Optimal control of fluids based on lattice Boltzmann method discretizations

Palma, Vryan Gil S.

Award date:
2008

Optimal Control of Fluids based on Lattice Boltzmann Method Discretizations

by

Vryan Gil S. Palma

Supervisor: Prof. Dr. René Pinnau

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Industrial Mathematics

Fachbereich Mathematik
Technische Universität Kaiserslautern

August 11, 2008

Abstract

In this study, the lattice Boltzmann method (LBM), based on the underlying statistical mechanics on assumed fluid particle population interacting over a discrete lattice mesh, is used to simulate fluid flows. LBM solves the macroscopic properties density, velocity and pressure of fluid modeled by the Navier-Stokes equations. Assuming a desired fluid behavior, we define an optimal control problem that has an objective function indicating the proximity of a simulated fluid behavior to the desired fluid behavior, and has the incompressible Navier-Stokes equations as the constraint. We aim to find the optimal control, in the form of an external volume force on fluid, that steers the fluid simulation towards the desired behavior. We perform the discrete adjoint method, a technique to determine the gradient of the objective function efficiently, that utilizes the LBM as a discrete solver of the constraint incompressible Navier-Stokes equations of the optimal control problem. Finally, we apply the method of steepest descent, an iterative method that uses the computed gradient, to search for the control that drives the simulation to optimality.

Acknowledgements

This work is the culmination of my 2-year European adventure as an Erasmus Mundus master student. I would like to extend my deepest gratitude to the administrators of the master programme in TU Eindhoven, the Netherlands and TU Kaiserslautern, Germany for taking care of us - the international master students.

I would like to thank our professors in TU Eindhoven for giving us the first trainings into what is it like being a master student. I would like to thank our professors in TU Kaiserslautern for shaping us further, making us improve and become better. I would like to thank Prof. Klar for his excellent teaching of the Numerics in PDE courses, Dr. Marheineke for an insightful experience working on the modeling seminar, Prof. Pinnau for his supervision in my master thesis and imparting on me how to work independently and Dr. Thömmes for his guidance in this work and making me appreciate the field of Computational Fluid Dynamics.

I would like to express my heartfelt gratitude to my Filipino family and friends who I can not thank enough for always being there for me even when we are so many miles apart. Finally, to the friends I met in Europe who, like me, have embarked into this wonderful journey, who I call my family here, I will be forever grateful for being part of this chapter of my life and for allowing me to be part of yours.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

To control a behavior of a system, as explained in a simple manner, pertains to influencing the outputs of the system by manipulating its inputs and be able to steer the system to produce desired outputs. This very idea opens up to a vast field of mathematics involving control problems, control techniques and their various applications. Among the many control approaches there are, optimal control is a control strategy where the control inputted enables the system to optimize a particular gauge or indication of a system's performance.

We shall be interested in optimal control on fluids and attempt to influence the behavior of fluid flows. Encompassing the small scale to the large scale, the blood circulating our veins, the milk one spilled on the table, those 6 or 7 tropical cyclones hitting the Philippines per year, in each day of our lives, we encounter countless occurrences characterized by fluid flow phenomena. Thus it is not surprising at all why efforts have been devoted to model fluid flow and undertakings have been directed to affect or alter fluid behavior. Causes that motivate studies on control on fluid flows include improving efficiency of production and product design, health, public safety, etc.

The Navier-Stokes equations mathematically describe fluid flows. It is an established set of equations that models a large number of flow phenomena of various interests. The Navier-Stokes equations, prevalently well-known as the foundation of many computational fluid dynamics problems, are derived from the conservation of mass, momentum, and energy and consequently give information regarding the macroscopic fluid properties: flow velocity, pressure, and density of fluid.

A conventional and familiar approach of solving the Navier-Stokes equations and its simplifications, such as the incompressible Navier-Stokes equations, is based on discretizing

the spatial domain to form a volume mesh or grid, transform the set of equations into its discretized form and then apply a suitable numerical scheme to solve the discretized equations. Discretization methods such as finite difference method (FDM), finite element method (FEM) and finite volume method (FVM) are employed alongside many other treatments and schemes that take care of efficient handling of discontinuities, avoidance of spurious oscillations and several other occurrences causing deviations to the accuracy of the fluid simulation.

On the other hand, one can choose alternative means of doing so. The lattice Boltzmann method (LBM) is a method based on underlying statistical mechanics on assumed fluid particle population interacting over a discrete lattice mesh. In this method based on microscopic models of fluid particles, the collective behavior of microscopic particles leads to the macroscopic properties of the fluid. And thus, with the LBM one solves an equivalent discrete mesoscopic system on a lattice structure instead of solving the macroscopic system. Among the primary scope of the applications of the LBM in computational fluid dynamics is the numerical solution of the incompressible Navier-Stokes equation. In this work, we shall put particular attention in involving the LBM in modeling fluid flows and solving the macroscopic properties velocity, pressure, and density.

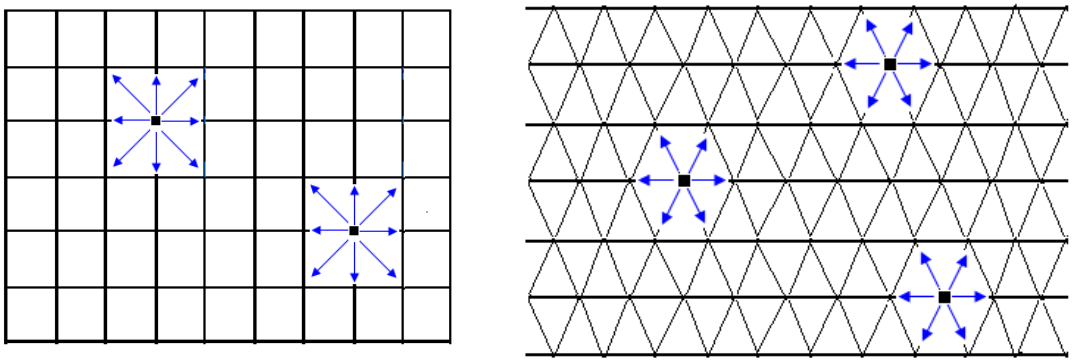

Figure 1.1: Examples of lattice mesh and indicated possibilities of particle travel directions

In the optimization that we shall be performing, first, we envision a certain fashion to how we desire the simulated fluid to behave. Our goal is to determine the optimal control, represented by an external volume force on the fluid, that maneuvers a simulation towards the desired behavior. Optimality will be measured by how the macroscopic properties of the simulated flow approximate those of the desired. We note that the macroscopic properties of the flow are the solutions of Navier-Stokes equations. In this work, we shall use the LBM to determine the macroscopic properties of the fluid flow. Hence, we shall use the LBM as a solver for the Navier-Stokes equations.

Having introduced the ingredients we need, we conduct an explorative study entitled *Optimal Control of Fluids based on Lattice Boltzmann Method Discretizations*. This work is organized as follows. In Chapter 2, we give an overview of the LBM and its fundamentals and present some fluid simulations using this method. We formally define in Chapter 3 an optimal control problem and discuss strategies in solving it. We also emphasize in the chapter the particular approach, the so-called "discretize and optimize," that we shall be heading to. In Chapter 4, we perform a systematic search for the optimal control among an infinite number of possible candidates and layout and analyze results. Finally, we wrap-up in Chapter 5 with our conclusions.

Chapter 2

The Lattice Boltzmann Method

Noted as an effective and promising tool in computational fluid dynamics, the lattice Boltzmann method is considered an alternative way of simulating fluid behavior apart from the conventional approaches of modeling and simulating fluid flows [4]. LBM is a method based on microscopic models of fluid particles and corresponding mesoscopic kinetic equations where ultimately the macroscopic properties of the fluid results from the collective behavior of the microscopic particles. LBM is a method that is characterized by the evolution of particle distributions and this evolution is basically composed of 1. a streaming process describing how the particles propagates in a phase space or velocity space wherein a simplified set of velocities is utilized, and 2. a collision process representing the redistribution of particle populations when the particles interact. Among the many strengths of this method, LBM has attracted attention due to its ability to easily represent complex physical phenomena, such as handling complex geometries and boundaries and incorporating interfacial interactions, simplicity in implementation and numerical efficiency [13], [4].

In this chapter, we present a brief overview of the LBM. We discuss the beginnings of the LBM from the theory of gases. We present a derivation of the LBM from the Boltzmann equation taking a 2-dimensional LBM model for demonstration. From this 2D LBM model, we derive the 2D incompressible Navier-Stokes equation. We then use the LBM model to simulate fluid flows and discuss results.

2.1 Origins: The lattice gas automata

The origins of the lattice Boltzmann method can be traced to the lattice gas automata (LGA). LGA is a dynamic that uses discrete time, space and particle velocities representing an evolution modeled by

$$n_i(\mathbf{x} + \mathbf{c}_i\Delta t,\ t + \Delta t) \ = \ n_i(\mathbf{x}, t) \ + \ \Omega_i(n(\mathbf{x}, t)), \tag{2.1}$$

where $\mathbf{x}$ refers to nodes that denotes a particle position in the phase space, t is a point in time and Δt is time increment. The variables n_i refers to a set of Boolean variables that describes the particle occupation at nodes and is associated with a number of particle velocities labeled $i = 1, \ldots, M$. The evolution of the particles consists of streaming and collision. Streaming refers to the movement of a particle to the nearest node dictated by the velocity $\mathbf{c}_i$, while collision refers to the rule of scattering and changing of position upon interaction of particles arriving at the same node. The said collision rule is represented by $\Omega_i(n(\mathbf{x}, t))$.

LBM differs from LGA since instead of Boolean variables n_i representing particle occupation at nodes, LBM uses a single-particle distribution. This particle distribution denotes the density or the amount per unit volume of particles being located about node $\mathbf{x}$ moving in velocity direction i at time t. LBM benefits from this replacement due to the elimination of the statistical noise in the method. Furthermore, a linearized collision operator in place of $\Omega_i(n(\mathbf{x}, t))$ is also used for simplification. A standard linearized collision operator is the well-known Bhatnagar-Gross-Krook (BGK) collision operator which uses a single-relaxation-time. In lattice BGK collision operator, a local equilibrium distribution term is present and this is chosen in such a way that we recover the Navier-Stokes equations. All in all, the derivation starts by replacing n in LGA equation (2.1) by f, the particle distribution, hence obtaining a kinetic equation for the particle distribution - the lattice Boltzmann equation (LBE). Its components is computed particularly the collision term so that it ultimately lead to the recovery of the macroscopic Navier-Stokes equation (see [4] for discussion.) We have

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

where similarly, $\mathbf{x}$ gives the particle position in the phase space, t is a point in time and Δt is time increment and the f_i's are the particle distributions associated with the particle velocities $\mathbf{c}_i$ with directions labeled $i = 1, \ldots, M$, respectively. $\Omega_i(f(\mathbf{x}, t))$ denotes the collision operator that represents the rate of change of f_i whenever particles collide. The particle velocity moments of f_i gives the macroscopic density ρ and momentum $\rho\mathbf{u}$, i.e.

$$\rho \ = \ \sum_i f_i,$$

$$\rho \mathbf{u} = \sum_i \mathbf{c}_i f_i,$$

$$p = c_s^2 \rho = \frac{1}{3}\rho.$$

2.2 From the Boltzmann equation to LBE

Formulated by Ludwig Boltzmann, the Boltzmann equation, also known as the Boltzmann transport equation, describes the evolution of a particle distribution density in a velocity space [22]. In the works [9] and [4], it has been shown that the LBE can also be obtained from the Boltzmann equation for discrete velocities by using a small Mach number expansion. From this derivation, LBE can be viewed as a discretized form of the Boltzmann equation wherein we perform the discretization in some manner in time and space. This way of derivation thus demonstrates that LBE is independent of LGA in contrast with what Section 2.1 suggests. We consider the Boltzmann equation equipped with Bhatnagar-Gross-Krook collision operator and let this single-relaxation-time approximation be our starting point. Hence, we have

$$\frac{\partial f}{\partial t} + \mathbf{c} \cdot \nabla f = -\frac{1}{\lambda}(f - g), \tag{2.2}$$

where $f \equiv f(\mathbf{x}, \mathbf{c}, t)$ is the single-particle distribution function, $\mathbf{c}$ is the microscopic velocity, λ is the relaxation time due to collision, and g is the Maxwell-Boltzmann distribution function. The Maxwell-Boltzmann distribution, a well-known distribution in statistical physics, gives the particle distribution of molecules in thermodynamic equilibrium [22] and is given by

$$g \equiv \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{(\mathbf{c}-\mathbf{u})^2}{2RT}\right),$$

where R is the ideal gas constant, D is the dimension of the space, and $\rho, \mathbf{u}$ and T are the macroscopic mass, velocity and temperature, respectively, and are the microscopic velocity moments of the distribution function f which obeys the conservation constraints

$$\rho = \int f \mathrm{d}\mathbf{c} = \int g \mathrm{d}\mathbf{c}, \tag{2.3}$$

$$\rho \mathbf{u} = \int \mathbf{c} f \mathrm{d}\mathbf{c} = \int \mathbf{c} g \mathrm{d}\mathbf{c}, \tag{2.4}$$

$$\rho \varepsilon = \frac{1}{2}\int (\mathbf{c}-\mathbf{u})^2 f \mathrm{d}\mathbf{c} = \frac{1}{2}\int (\mathbf{c}-\mathbf{u})^2 g \mathrm{d}\mathbf{c}, \tag{2.5}$$

where the temperature can be derived using the relation involving the number of degrees of freedom of a particle D_0, Avogadro's constant N_A and the Boltzmann constant k_B,

$$\varepsilon = \frac{D_0}{2} RT = \frac{D_0}{2} N_A k_B T.$$

Defining the material derivative or the time derivative along the characteristic line $\mathbf{c}$ as

$$\frac{\mathrm{d}}{\mathrm{d}t} \equiv \frac{\partial}{\partial t} + \mathbf{c} \cdot \nabla,$$

we can write (2.2) as

$$\frac{\mathrm{d}f}{\mathrm{d}t} + \frac{1}{\lambda} f = \frac{1}{\lambda} g. \tag{2.6}$$

We integrate (2.6) over a time step of Δt. Under the assumption that Δt is small enough and that g is locally smooth enough such that terms of order $O(\Delta t^2)$ can be neglected in its approximation, we obtain

$$f(\mathbf{x} + \mathbf{c}\Delta t, \mathbf{c},\ t + \Delta t) - f(\mathbf{x}, \mathbf{c}, t) = -\frac{1}{\tau} \left[f(\mathbf{x}, \mathbf{c}, t) - g(\mathbf{x}, \mathbf{c}, t) \right] \tag{2.7}$$

where $\tau \equiv \lambda / \Delta t$ is the dimensionless relaxation time. (2.7) is the evolution equation of the distribution function f with discrete time.

One can observe that g is a function of macroscopic variables $\rho, \mathbf{u}$ and T, hence, calculation of $\rho, \mathbf{u}$ and T is needed to construct the equilibrium function g. A properly constructed equilibrium function will lead to the Navier-Stokes equation [9], [18]. With the use of

$$\int \psi(\mathbf{c}) g(\mathbf{x}, \mathbf{c}, t) \mathrm{d}\mathbf{c} \ = \ \sum_i W_i \psi(\mathbf{c}_i) g(\mathbf{x}, \mathbf{c}_i, t)$$

as an appropriate discretization in the velocity space so that integration can be approximated by quadrature up to a certain degree of accuracy, where $\psi(\mathbf{c})$ is a polynomial function of $\mathbf{c}$, W_i is the weight coefficient of the quadrature and $\mathbf{c}_i$ is the discrete velocity, we can be able to discretize the conservation constraints (2.3)-(2.5) as

$$\rho \ = \ \sum_i f_i \ = \ \sum_i g_i,$$

$$\rho \mathbf{u} \ = \ \sum_i \mathbf{c}_i f_i \ = \ \sum_i \mathbf{c}_i g_i,$$

$$\rho \varepsilon \ = \ \frac{1}{2} \sum_i (\mathbf{c}_i - \mathbf{u})^2 f_i \ = \ \frac{1}{2} \sum_i (\mathbf{c}_i - \mathbf{u})^2 g_i,$$

where

$$f_i \equiv f_i(\mathbf{x}, t) \equiv W_i f(\mathbf{x}, \mathbf{c}_i, t), \tag{2.8}$$

$$g_i \equiv g_i(\mathbf{x}, t) \equiv W_i g(\mathbf{x}, \mathbf{c}_i, t). \tag{2.9}$$

As fully detailed in [9], we apply the low Mach number expansion on g and retain only the terms up to order $O(\mathbf{u}^2)$ and denote this as f^{eq}. We shall have

$$g \sim f^{\text{eq}} \equiv \frac{\rho}{(2\pi RT)^{D/2}} \exp\left(-\frac{\mathbf{c}^2}{2RT}\right) \times \left\{1 + \frac{(\mathbf{c}\cdot\mathbf{u})}{RT} + \frac{(\mathbf{c}\cdot\mathbf{u})^2}{2(RT)^2} - \frac{\mathbf{u}^2}{2RT}\right\}.$$

Further, it has been remarked in [9] that discretizing the momentum space is coupled to discretizing the configuration space such that a lattice structure is obtained. The quadrature must also be accurate enough to satisfy the conservation constraints and retain the necessary symmetries required to recover the Navier-Stokes equation.

In this work, we shall simulate 2-dimensional fluid flows and use the the lattice Boltzmann model in two-dimensional space with 9 velocities, also known as D2Q9. And hence, resulting from the discussions above, using the notations

$$\mathbf{c}_i = \begin{cases} (0,0), & \text{if } \alpha = 0, \\ \left(\cos\left[\frac{(\alpha-1)\pi}{2}\right],\ \sin\left[\frac{(\alpha-1)\pi}{2}\right]\right) c, & \text{if } \alpha = 1,2,3,4 \\ \left(\cos\left[\frac{(2\alpha-9)\pi}{4}\right],\ \sin\left[\frac{(2\alpha-9)\pi}{4}\right]\right) \sqrt{2}c, & \text{if } \alpha = 5,6,7,8. \end{cases}$$

and a set of weight coefficients

$$w_i = \begin{cases} \frac{4}{9}, & \text{if } \alpha = 0, \\ \frac{1}{9}, & \text{if } \alpha = 1,2,3,4 \\ \frac{1}{36}, & \text{if } \alpha = 5,6,7,8, \end{cases}$$

obtained from W_i in (2.8)-(2.9) and given by the relation

$$W_i = (2\pi RT)^{D/2} \exp\left(\frac{\mathbf{c}^2}{2RT}\right) w_i,$$

with $RT = c_s^2 = c^2/3$, the equilibrium distribution f_i^{eq} for the 9-velocity lattice Boltzmann evolution equation

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t,\ t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[f_i(\mathbf{x}, t) - f_i^{\text{eq}}(\mathbf{x}, t)\right] \tag{2.10}$$

is given by

$$f_i^{\text{eq}} = w_i \rho \left[1 + 3\frac{(\mathbf{c}_i\cdot\mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i\cdot\mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right]. \tag{2.11}$$

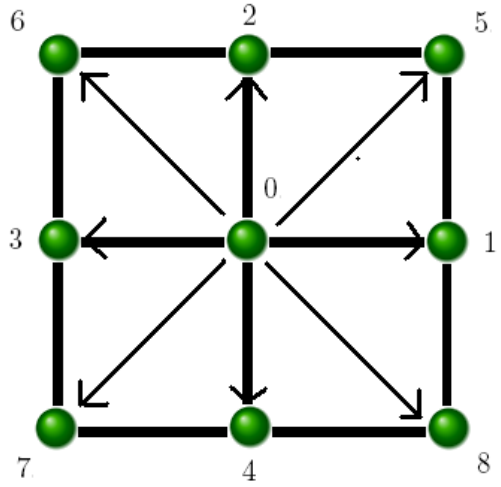

Figure 2.1: The discrete 9-velocity set labeled $i = 0, \ldots, 8$ on a square lattice in two dimensions

2.3 From LBE to the incompressible Navier-Stokes equations

There have been several works and several treatments applied to demonstrate the recovery of the Navier-Stokes equation from the LBE such as those in [8], [12], [18]. Since in this work, the incompressible Navier-Stokes equations shall be of interest, we follow first a proposed lattice Boltzmann model for incompressible Navier-Stokes equations as presented in [8]. The incompressible Navier-Stokes equations are then derived from the proposed incompressible lattice Boltzmann model via Chapman-Enskog procedure.

[8] discusses and sets forth a necessity to improve the LBM for simulations of the incompressible Navier-Stokes equations in general, especially for unsteady flows, due to the compressible effect, a trait of LBM causing density fluctuation. The work develops a lattice Boltzmann model for the incompressible Navier-Stokes equations where the fundamental idea suggests that the compressible effect can be explicitly eliminated by discarding the terms of higher order Mach number, i.e. the terms of order $O(\mathrm{Ma}^2)$.

In an incompressible flow, ideally, so that the divergence of the velocity is zero, the density is assumed as constant, i.e. $\rho = \rho_0$. Assuming an order $O(\mathrm{Ma}^2)$ density fluctuation of $\delta\rho$,

we write $\rho = \rho_0 + \delta\rho$ and substituting to (2.11), we have

$$\begin{aligned} f_i^{\text{eq}} &= w_i(\rho_0 + \delta\rho)\left[1 + 3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right] \\ &= w_i\left[(\rho_0 + \delta\rho) + \rho_0\left(3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right)\right. \\ &\qquad \left. + \delta\rho\left(3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right)\right]. \end{aligned}$$

And since $\text{Ma} \equiv \frac{u}{c}$, ignoring $O(\text{Ma}^3)$ terms or higher such as $\delta\rho \cdot \frac{\mathbf{u}}{c}$ and $\delta\rho \cdot \frac{\mathbf{u}^2}{c^2}$, we obtain

$$f_i^{\text{eq}} = w_i\left[\rho + \rho_0\left(3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right)\right]. \tag{2.12}$$

This is the equilibrium distribution function of the incompressible lattice Boltzmann model where terms are of order $O(\text{Ma}^2)$ or lower. Since p typically appears in the incompressible Navier-Stokes equation as an independent variable, we introduce a local pressure distribution defined as

$$p_i \equiv c_s^2 f_i = \frac{1}{3} f_i,$$

and hence from here, the p-representation of the evolution equation of the LBE system is obtained and given by

$$p_i(\mathbf{x} + \mathbf{c}_i\Delta t,\ t + \Delta t) - p_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[p_i(\mathbf{x}, t) - p_i^{\text{eq}}(\mathbf{x}, t)\right]. \tag{2.13}$$

And since $p = \frac{1}{3}\rho$ (and $p_0 = \frac{1}{3}\rho_0$),

$$p_i^{\text{eq}} \equiv c_s^2 f_i^{\text{eq}} = w_i\left[p + p_0\left(3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right)\right] \tag{2.14}$$

and the macroscopic properties are given by

$$p = \sum_i p_i, \tag{2.15}$$

$$p_0\mathbf{u} = \sum_i \mathbf{c}_i p_i, \tag{2.16}$$

$$\rho = \frac{1}{c_s^2} p = 3p. \tag{2.17}$$

Hence, (2.13)-(2.17) comprise the proposed lattice Boltzmann model for incompressible Navier-Stokes equations in [8] we shall be adapting in this work.

At this point, through the proposed model, the incompressible Navier-Stokes equation can be derived via the Chapman-Enskog procedure. The Chapman-Enskog procedure provides

a way of relating the mesoscopic picture of the Boltzmann equation to the macroscopic hydrodynamic picture by successive aprroximations using a mean-free timescale as a perturbation parameter [20]. Since a thorough discussion of the Chapman-Enskog analysis goes beyond the scope of our study, we only present in following an outline of the analysis based on [8] where the full details can be found.

We introduce the following expansions

$$\begin{aligned} f_i(\mathbf{x}+\mathbf{c}_i\Delta t,\ t+\Delta t) &= \sum_{n=0}^{\infty}\frac{\varepsilon^n}{n!}D_t^n f_i(\mathbf{x},t) = f_i + D_t f_i + \frac{\varepsilon^2}{2!}D_t^2 f_i + \ldots, \\ f_i &= \sum_{n=0}^{\infty}\varepsilon^n f_i^{(n)} = \varepsilon^0 f_i^{(0)} + \varepsilon^1 f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \ldots, \\ \partial_t &= \sum_{n=0}^{\infty}\varepsilon^n \partial_{t_n} = \varepsilon^0\partial_{t_0} + \varepsilon^1\partial_{t_1} + \varepsilon^2\partial_{t_2} + \ldots, \end{aligned}$$

where $\varepsilon = \Delta t$ and $D_t \equiv (\partial_t + \mathbf{c}_i\cdot\nabla)$. We can rewrite the lattice Boltzmann equation (2.10)

$$\text{to order of } O(\varepsilon^0): \qquad f_i^{(0)} = f_i^{\mathrm{eq}}, \tag{2.18}$$

$$\text{to order of } O(\varepsilon^1): \qquad D_{t_0} f_i^{(0)} = -\frac{1}{\tau} f_i^{(1)}, \tag{2.19}$$

$$\text{to order of } O(\varepsilon^2): \qquad \partial_{t_1} f_i^{(0)} + \left(\frac{2\tau-1}{2\tau}\right) D_{t_0} f_i^{(1)} = -\frac{1}{\tau} f_i^{(2)}, \tag{2.20}$$

where f_i^{eq} is defined in (2.12) and $D_{t_n} \equiv (\partial_{t_n} + \mathbf{c}_i\cdot\nabla)$. The conservation constraints can be written as

$$\begin{aligned} \sum_i f_i^{(0)} &= \rho, \\ \sum_i \mathbf{c}_i f_i^{(0)} &= \rho_0\mathbf{u}, \\ \sum_i f_i^{(n)} &= 0, \quad \forall n > 0, \\ \sum_i \mathbf{c}_i f_i^{(n)} &= 0, \quad \forall n > 0. \end{aligned}$$

The Chapman-Enskog ansatz states that the time dependence of f_i is through the variables ρ and $\mathbf{u}$ and hence

$$\partial_t f_i = \frac{\partial f_i}{\partial \rho}\partial_t\rho + \frac{\partial f_i}{\partial \mathbf{u}}\partial_t\mathbf{u}.$$

Through the properties of the 9-bit tensor $E^{(n)} = \sum_{i\neq 0} w_i \mathbf{c}_{i,1}\mathbf{c}_{i,2}\ldots\mathbf{c}_{i,n}$, we can obtain

$$\sum_i f_i^{(0)} = \rho, \tag{2.21}$$

$$\sum_i \mathbf{c}_i f_i^{(0)} = \rho_0 \mathbf{u}, \tag{2.22}$$

$$\sum_i \mathbf{c}_{i,\alpha}\mathbf{c}_{i,\beta} f_i^{(0)} = \frac{1}{3}c^2\rho\delta_{\alpha\beta} + \rho_0 u_\alpha u_\beta, \tag{2.23}$$

$$\sum_i \mathbf{c}_{i,\alpha}\mathbf{c}_{i,\beta}\mathbf{c}_{i,\gamma} f_i^{(0)} = \frac{1}{3}c^2\rho_0(\delta_{\alpha\beta}u_\gamma + \delta_{\gamma\alpha}u_\beta + \delta_{\beta\gamma}u_\alpha). \tag{2.24}$$

The moments of (2.19) lead to the zeroth-order results

$$\partial_{t_0}\rho + \nabla\cdot(\rho_0\mathbf{u}) = 0 \tag{2.25}$$

$$\partial_{t_0}(\rho_0\mathbf{u}) + \nabla\cdot\Pi^{(0)} = 0, \tag{2.26}$$

where $\Pi^{(0)} = \sum_i \mathbf{c}_i\mathbf{c}_i f_i^{(0)}$ is the zeroth-order momentum flux-tensor. Using $\Pi^{(0)}_{\alpha\beta}$ given in (2.23), (2.25)-(2.26) can be written as

$$\begin{aligned}\frac{1}{c_s^2}\partial_{t_0}\tilde{p} + \nabla\cdot\mathbf{u} &= 0\\ \partial_{t_0}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} &= -\nabla\tilde{p} - \mathbf{u}\nabla\cdot\mathbf{u},\end{aligned}$$

where $\tilde{p} = c_s^2\rho/\rho_0$ is the normalized pressure.

The moments of (2.20) lead to the first-order results

$$\begin{aligned}\partial_{t_1}\rho &= 0\\ \partial_{t_1}(\rho_0\mathbf{u}) + \left(\frac{2\tau-1}{2\tau}\right)\nabla\cdot\Pi^{(1)} &= 0,\end{aligned}$$

where $\Pi^{(1)} = \sum_i \mathbf{c}_i\mathbf{c}_i f_i^{(1)}$ which can be expanded into

$$\Pi^{(1)}_{\alpha\beta} = \tau\left[c_s^2(u_\alpha\rho + u_\beta\nabla_\alpha\rho) + \rho_0(u_\alpha(\mathbf{u}\cdot\nabla)u_\beta + u_\beta(\mathbf{u}\cdot\nabla)u_\alpha) - c_s^2\rho_0(\nabla_\alpha u_\beta + \nabla_\beta u_\alpha)\right].$$

We discard terms of order $O(\mathrm{Ma}^3)$ and terms of order $O(\mathrm{u}^3)$ to be consistent with the small velocity expansion of f_i^{eq} up to the order $O(\mathrm{u}^2)$ resulting to

$$\left(\frac{2\tau-1}{2\tau}\right)\nabla_\alpha\cdot\Pi^{(1)}_{\alpha\beta} = -\frac{\nu}{\Delta t}\rho_0\nabla^2 u_\alpha + O(\mathrm{Ma}^3),$$

where

$$\nu = \frac{2\tau-1}{6}\frac{\Delta x^2}{\Delta t}.$$

Combining the zeroth and first order results where $\partial_t = \partial_{t_0} + \varepsilon\partial_{t_1}$, setting expansion parameter $\varepsilon = 1$, we obtain the incompressible Navier-Stokes equation accurate to order $O(\mathrm{Ma}^2)$ for the continuity equation and $O(\mathrm{Ma}^3)$ in momentum equation

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0 + O(\mathrm{Ma}^2) \\ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} &= -\nabla \tilde{p} + \nu \Delta \mathbf{u} + O(\mathrm{Ma}^3). \end{aligned}$$

2.4 Boundary Conditions

The problem of formulating the proper boundary conditions to be used in the LBM model comes down to finding the most suitable relations between the incoming and outgoing populations [21]. An important consideration in the modeling of fluid flows is the treatment of the fluid surroundings. Normally, macroscopic information on boundaries are given such as no-slip boundary conditions on walls for fluid flows. Translating this macroscopic information into the microscopic distribution functions is where the difficulties arises. Different treatments are proposed by different authors as these treatments are developed to provide better accuracy and stability for the simulation [13].

We shall examine two classes of boundary conditions in this work, namely, the periodic boundary conditions and the no-slip boundary conditions and apply it on a simple domain represented by a rectangular grid.

2.4.1 Periodic Boundary Conditions

Naturally, when one wants to model a flow on an infinite domain, one takes a finite portion of the domain and apply periodic boundary conditions on the domain boundaries. The main idea is that a traveling particle which go outs of the system on the right will reenter on the left. Periodic boundary conditions replicate the behavior of the fluid making a simulation consisting only of a small lattice but behaving as if it is infinite in size. Periodic boundary conditions are typically used when the effects of the surface can be neglected.

To implement the periodic boundary conditions, we use the concept of buffers. This means we associate a buffer to a lattice node and store temporarily in memory the information such as particle distribution, density, velocity, pressure, etc. for each lattice node while the information moves from one place to another due to either streaming or collision. As illustrated in Figure 2.2, 1 represents the fluid particle nodes and 0 as a ghost node represents either the walls as boundary or the flow inlet or outlet in a domain. We have

an $m \times n$ lattice where there are $m-2$ fluid particle nodes along the vertical axis and $n-2$ fluid particle nodes along the horizontal axis.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Figure 2.2: 12×7 Square Lattice Structure

Consider the D2Q9 lattice Boltzmann model in two-dimensional space with 9 velocities. Let us examine what happens to the particle distribution f_i (or p_i for the proposed model of [8]) associated with each of the 9 velocity directions as labeled in Figure 2.1. Here, we use 9 matrices f_i where particle distribution is stored at each lattice node. Upon application of the streaming process to each matrix depending on the associated velocity direction, the periodic boundary condition is imposed by filling the empty inward population by the outward population. As an illustration, Figure 2.3 depicts the streamed population along direction 1. As described, we fill the empty incoming population by the outgoing population which are temporarily lying on the ghost nodes. Finally, 0 is reassigned to ghost nodes. Imposing periodicity for the streamed population along the other directions follows in similar fashion.

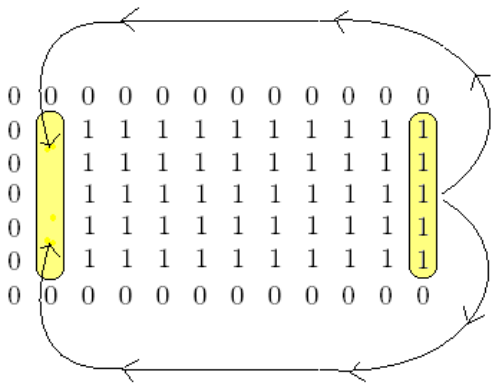

Figure 2.3: Streaming process along direction $i = 1$ employing periodic boundary conditions on the flow inlet and outlet

2.4.2 No-slip Boundary Conditions

One of the most typically used boundary condition is the no-slip boundary condition. It is a familiar knowledge that a no-slip boundary condition for viscous fluid represents a fluid with zero velocity at a solid boundary. The idea behind this states that at the contact surface, a moving fluid in contact with a solid body will not have any velocity relative to the body. For the implementation of the no-slip boundary conditions, the so-called "bounce-back" treatment is typically used. A simple [24] approach is a direct reflection of particles approaching the walls. Here, when a particle hits a node of the wall, the particle distribution scatters back to the node it came from. Let us apply the bounce-back boundary conditions in the D2Q9 model as illustrated in Figure 2.4. Consider the particles that arrives at nodes W, X, Y and Z. Whenever a particle going in direction 8 arrives at W from node X, a direction-6 particle is bounced back to node X the following time step. Similarly, if a particle traveling along direction 4 arrives at W from Y, a direction-2 particle is sent back to node Y the following time step, and for a particle traveling along direction 7 arrives W, a direction-5 particle is sent back to node Z. This logic using direct reflection

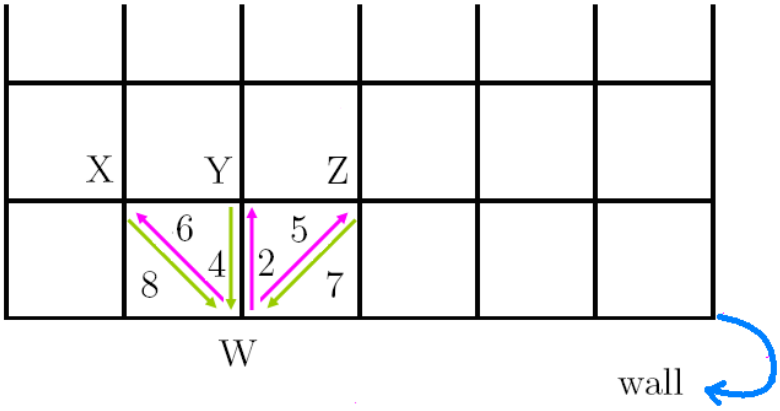

Figure 2.4: The bounce-back boundary condition

at the walls as implemented with use of buffers is shown in Figure 2.5. Here, the figure in the left-hand side illustrates the population streamed along direction 2 and in the figure in the right-hand side the population streamed along direction 4. In the left-hand side figure, the population arriving at the wall is bounced back to its original location before the streaming along the opposite direction, i.e. direction 4. The same concept applies to the population arriving at the wall in the right-hand side figure.

It is worth pointing out that there are several variations and improvements to the simple bounce-back technique that is pointed above as presented in various works. In [11], some approaches for the implementation of no-slip boundary condition were compared and a

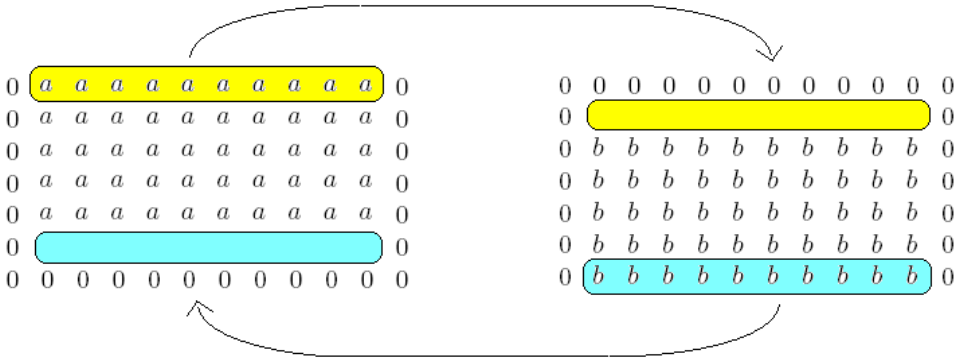

Figure 2.5: Implementation of bounce-back boundary condition in the streaming process

developed approach is proposed along with the advantages and difficulties that comes with it.

2.5 Implementation of LBM and simulations of fluid

In the previous sections, we provide necessary background as foundation to be able to implement LBM and simulate fluid flow. Summarizing, as we have seen in the previous sections the lattice Boltzmann evolution equation using the simplest collision operator, i.e. the single-relaxation-time BGK, is given by

$$f_i(\mathbf{x} + \mathbf{c}_i\Delta t,\ t + \Delta t) - f_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[f_i(\mathbf{x}, t) - f_i^{\mathrm{eq}}(\mathbf{x}, t)\right]$$

with the equilibrium distribution

$$f_i^{\mathrm{eq}} = w_i\rho\left[1 + 3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right].$$

and equations for the macroscopic properties

$$\rho = \sum_i f_i, \quad \rho\mathbf{u} = \sum_i \mathbf{c}_i f_i, \quad p = \frac{1}{3}\rho. \tag{2.27}$$

The right-hand side of the evolution equation represents the collision process while the left-hand side represents the streaming. Furthermore, (2.27) gives the update rule for the macroscopic properties at each point in time in the evolution.

As we expect a constant density for the incompressible Navier-Stokes equation, we use a corresponding formulation, setting $p = c_s\rho$ and $p_0 = c_s\rho_0$, with the evolution equation is

given by

$$p_i(\mathbf{x} + \mathbf{c}_i\Delta t,\ t + \Delta t) - p_i(\mathbf{x}, t) = -\frac{1}{\tau}\left[p_i(\mathbf{x}, t) - p_i^{\text{eq}}(\mathbf{x}, t)\right]$$

with the equilibrium pressure distribution

$$p_i^{\text{eq}} \equiv c_s^2 f_i^{\text{eq}} = w_i\left[p + p_0\left(3\frac{(\mathbf{c}_i \cdot \mathbf{u})}{c^2} + \frac{9}{2}\frac{(\mathbf{c}_i \cdot \mathbf{u})^2}{c^4} - \frac{3}{2}\frac{\mathbf{u}^2}{c^2}\right)\right]$$

and equations for the macroscopic properties

$$p\ =\ \sum_i p_i, \qquad p_0\mathbf{u}\ =\ \sum_i \mathbf{c}_i p_i, \qquad \rho = 3p$$

where the pressure $\tilde{p}$ appearing in the Navier-Stokes equation is given by $\tilde{p} = \dfrac{p}{\rho_0}$ and the kinematic viscosity is $\nu = \dfrac{2\tau - 1}{6}$. In addition, this model offers computational simplicity that shall benefit us in the succeeding chapters wherein we evaluate derivatives.

For our fluid simulation, it is an important point to emphasize our interest to apply an external force to drive the fluid flow. This external force shall then be regarded as a control that we shall be investigating further towards the next chapters. To incorporate an external body force to engine the fluid flow, we introduce a simple force term of the form

$$\frac{1}{3}\mathbf{c}_i \cdot \mathbf{F}$$

where $\mathbf{F} = (F_x, F_y)^T$ is a vector representing the 2-dimensional volume force and shall be introduced as a part of the collision term.

As already introduced in the previous section, for our implementation, we use buffers for storing particle distributions, density, velocity, pressure, etc. for each location. This technique provides an easy implementation of the streaming and the application of the boundary conditions as described in the previous section. Note that there are many ways to carry out the implementation of LBM where one normally takes into account the computational speed and efficiency of the implementation in fulfilling the dictates of the application where it will be used. We start with initializing values for the distributions and macroscopic properties. Since the we are dealing with a evolutionary problem, we produce a sequence of iterates for each point in time and ideally, at some point, a steady-state solution is reached and the process is terminated. One possible condition to impose stipulating that the steady-state is reached is by using the criteria

$$\frac{\sum_i \sum_j \left(\left|u^{(x)}(\mathbf{x}_{ij}, t + \Delta t) - u^{(x)}(\mathbf{x}_{ij}, t)\right| + \left|u^{(y)}(\mathbf{x}_{ij}, t + \Delta t) - u^{(y)}(\mathbf{x}_{ij}, t)\right|\right)}{\sum_i \sum_j \left(\left|u^{(x)}(\mathbf{x}_{ij}, t)\right| + \left|u^{(y)}(\mathbf{x}_{ij}, t)\right|\right)}\ <\ \text{tol}, \tag{2.28}$$

where tol is an assigned tolerance value. We present here an algorithm that we shall use for the LBM as outlined in Algorithm 1.

Algorithm 1 D2Q9 LBM with BGK

Set initialization values:

$p_0 = p_{\text{init}}$
$\mathbf{u}_0 = \mathbf{u}_{\text{init}}$
$p_{i,0} = p_{i,0}^{\text{eq}}(p_0, \mathbf{u}_0)$
$t = 1$

while steady-state is not yet reached **do**

compute the equilibrium distribution
$p_{i,t}^{\text{eq}}(p_t, \mathbf{u}_t)$
perform collision
$\bar{p} = \left(1 - \frac{1}{\tau}\right) p_{i,t} + \frac{1}{\tau} p_{i,t}^{\text{eq}}(p_t, \mathbf{u}_t)$
apply forcing
$\bar{p} = \bar{p} + F$
perform streaming
apply boundary condition technique
update
$p_{i,t+1} = \bar{p}$
$p_{t+1} = \sum_i p_{i,t+1}$
$\mathbf{u}_{t+1} = \sum_i c_i p_{i,t+1}$
$t = t + 1$

end while

Finally, using what we have gathered, we implement LBM to simulate fluid flows. All simulations presented here have been performed in Matlab 7.0.4 on a T60 IBM Thinkpad.

2.5.1 Flow on an infinite channel

First, let us consider a flow on an infinite channel driven by a constant volume force. We model this on the unit square domain assigning the left side to be the flow inlet and the right to be the flow outlet, and the top and the bottom to be the channel walls. We construct a square mesh in this domain for the lattice. To model the flow on an infinite

channel, we prescribe periodic boundary conditions on the inlet and the outlet. The no-slip boundary conditions will be modeled by the bounce-back scheme at the channel walls.

We run simulations for 25×30 lattice of fluid particles, kinematic viscosity of 1 (giving $\tau = 3.5$) and macroscopic density ρ_0 of 1 for every node. Time step Δt is assumed to be 1.

First we test the trivial case when the fluid has zero initial velocities $u_0^{(x)} = 0$ and $u_0^{(y)} = 0$ and volume force vector $\mathbf{F} = (0, 0)^T$ is applied. As expected, for any number of iterations, $u_t^{(x)}$ and $u_t^{(y)}$ remained zero, and p is 1 for every node.

Next, we investigate what happens when the fluid has nonzero initial velocity but there is an absence of force to drive the flow. We assign initial velocity $u_0^{(x)}$ to have a parabolic profile having a maximum value $u_{0,\max}^{(x)} = 1$ while $u_0^{(y)} = 0$ and volume force vector $\mathbf{F} = (0, 0)^T$ is applied. Figure 2.6 shows the evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 500$. The figure on the upper left plots $u_t^{(x)}$ for $t = 1, \ldots, 500$. The parabola marked by $+$ represents the initial horizontal velocity $u_0^{(x)}$, the parabola marked by $\circ$ gives $u_{50}^{(x)}$, $\triangleleft$ gives $u_{100}^{(x)}$, $\triangleright$ gives $u_{300}^{(x)}$ and $*$ for gives $u_{500}^{(x)}$. For each iterations, the parabola flattens until $u_t^{(x)}$ becomes zero. The flattening of the parabola can be attributed to the effect of the viscosity as the fluid approaches its steady-state. The particles have an initial horizontal velocity but the viscosity slows down the motion of the particles until they eventually stop moving. The figure on the upper right plots $u_t^{(y)}$ for $t = 1, \ldots, 500$. As shown, they are characterized by computational noise with magnitude of 10^{-16} suggesting that the vertical velocity that is initially set to zero remains zero. The figure on the bottom suggests p remained constant throughout $t = 1, \ldots, 500$. On both last 2 figures, $+$ represents the initial profile while $\times$ marks the profile at $t = 500$.

Next, we assign initial $u_0^{(x)}$ to have a parabolic profile having a maximum value $u_{0,\max}^{(x)} = 1$ while $u_0^{(y)} = 0$ and volume force vector $\mathbf{F} = (0.1, 0)^T$ is applied. Now the simulation is driven by a constant volume force eastwards. We observe the flow until it reaches the steady-state as described by (2.28) where we set tol $= 10^{-7}$, the machine single-precision. Figure 2.7 shows the steady-state $u_t^{(x)}$ reached after 637 iterations where $u_{637,\max}^{(x)} = 15.9989$. We observe that the particles in the center travels with the highest speed and the speed decreases when the particles are nearer the channel walls. Figure 2.8 shows the evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 637$. The figure on the upper left plots $u_t^{(x)}$ for $t = 1, \ldots, 637$. The parabola marked by $+$ represents the initial horizontal velocity $u_0^{(x)}$, the parabola marked by $\circ$ gives $u_{50}^{(x)}$, $\triangleleft$ gives $u_{100}^{(x)}$, $\triangleright$ gives $u_{300}^{(x)}$ and $*$ for gives $u_{637}^{(x)}$. The parabola colored red gives the analytical solution which gives an error difference of 0.0479 with respect to the obtained numerical solution.

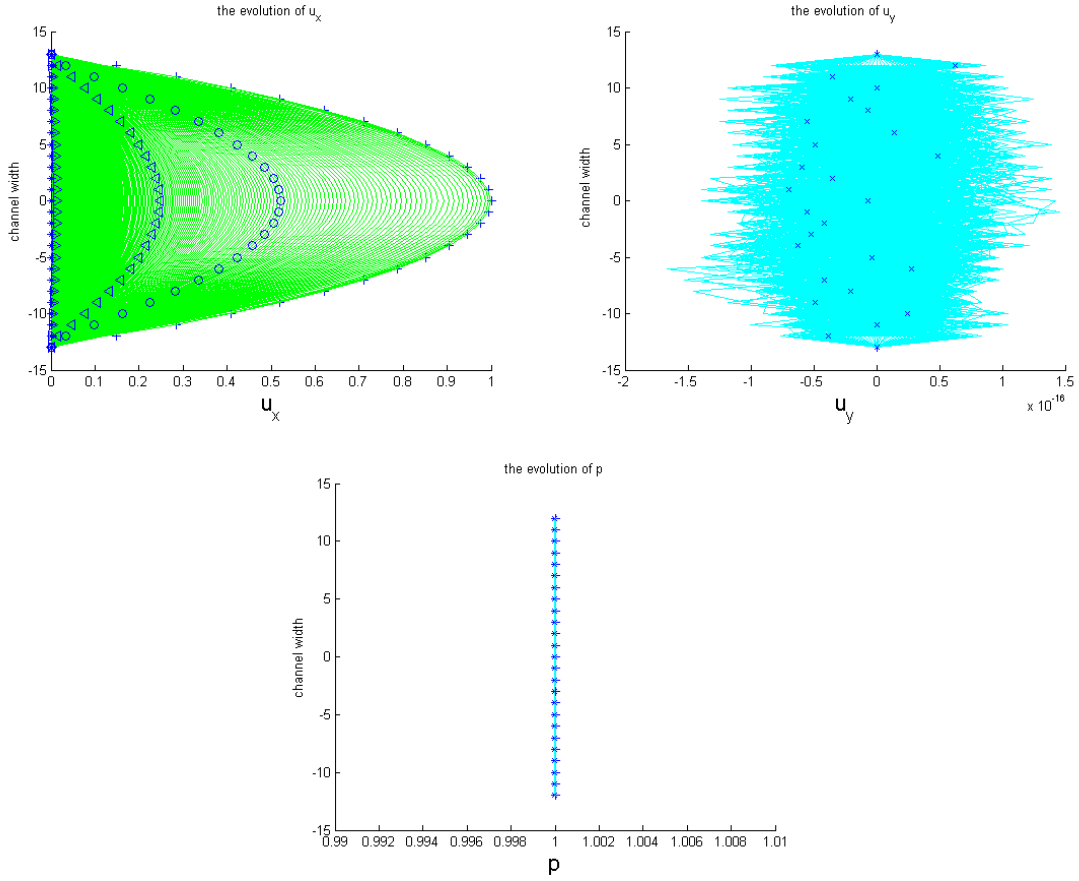

Figure 2.6: The evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 500$ assigning $u_0^{(x)}$ to have a parabolic profile with $u_{0,\max}^{(x)} = 1$, $u_0^{(y)} = 0$ and $\mathbf{F} = (0,0)^T$

As we observe, the forcing increases the initial velocity for some time until the velocity converge to the steady-state velocity. As shown in the upper right, $u_t^{(y)}$ for $t = 1, \ldots, 637$ are characterized by computational noise with magnitude of 10^{-14} suggesting that the vertical velocity remains zero. The figure on the bottom suggests p remained constant throughout $t = 1, \ldots, 637$.

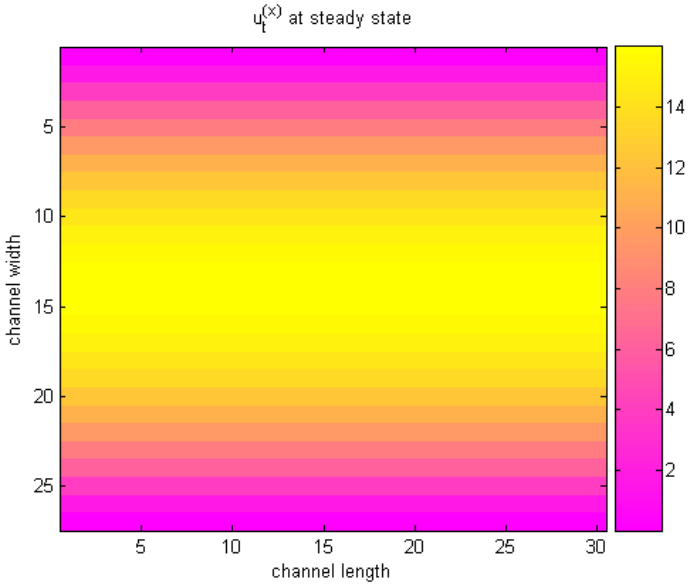

Figure 2.7: The $u_t^{(x)}$ at the steady-state at $t = 637$

We also investigate and test the effect of varying initial velocities by fixing the kinematic viscosity and forcing. Figure 2.9 shows results for fluid kinematic viscosity of 1, volume force vector $\mathbf{F} = (0.1, 0)^T$, setting $u_0^{(y)} = 0$ for 3 cases of initial horizontal velocity $u_t^{(x)}$ shown in the left figure: 1. for $u_0^{(x)}$ having a parabolic profile with $u_{0,\max}^{(x)} = 1$ marked by $\triangleleft$, 2. for $u_0^{(x)} = 0$ marked by $+$ and 3. for $u_0^{(x)}$ having a parabolic profile which is negative of case 1 marked by $\circ$. The figure on the right suggests the coinciding of the resulting steady-state horizontal velocities obtained marked by $\triangleleft, +$ and $\circ$ after 637, 641 and 646 iterations, respectively for each cases. All steady-state horizontal velocities obtained have $u_{t,\max}^{(x)} = 15.9989$.

Lastly, we compare the effect of the varying the viscosity in reaching the steady-state. We set the volume force vector to be $\mathbf{F} = (0.1, 0)^T$, $u_0^{(x)}$ to have a parabolic profile with $u_{0,\max}^{(x)} = 1$ and $u_0^{(y)} = 0$. Table 2.1 indicates that larger viscosity implies less iterations to reach the steady-state and lower horizontal velocity at the steady-state.

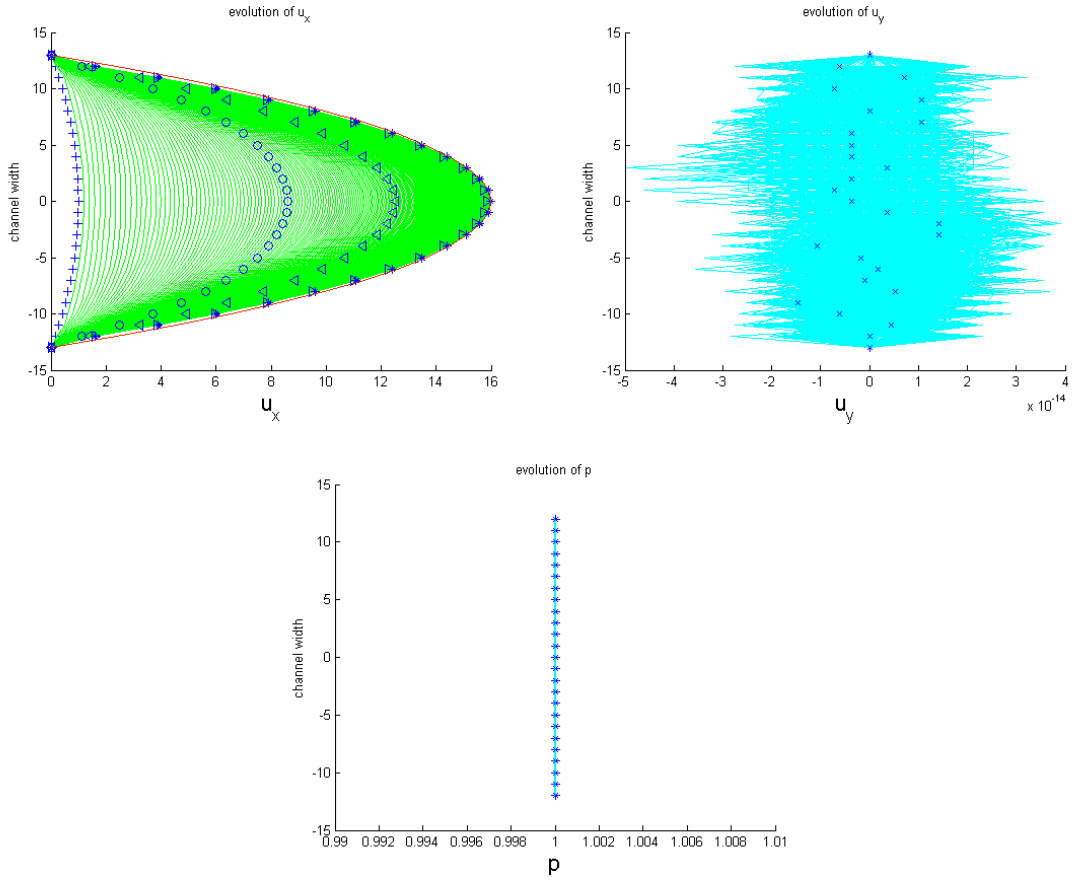

Figure 2.8: The evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 637$ assigning $u_0^{(x)}$ to have a parabolic profile with $u_{0,\max}^{(x)} = 1$, $u_0^{(y)} = 0$ and $\mathbf{F} = (0.1, 0)^T$

viscosity μ	τ	no. of iterations	$u_{t,\max}^{(x)}$
0.75	2.75	807	21.0481
1	3.5	637	15.9989
1.5	5	473	11.0994
2	6.5	407	8.7996
2.25	7.25	393	8.0830

Table 2.1: Comparison of the effect of varying the viscosity in reaching the steady-state for flow on an infinite channel

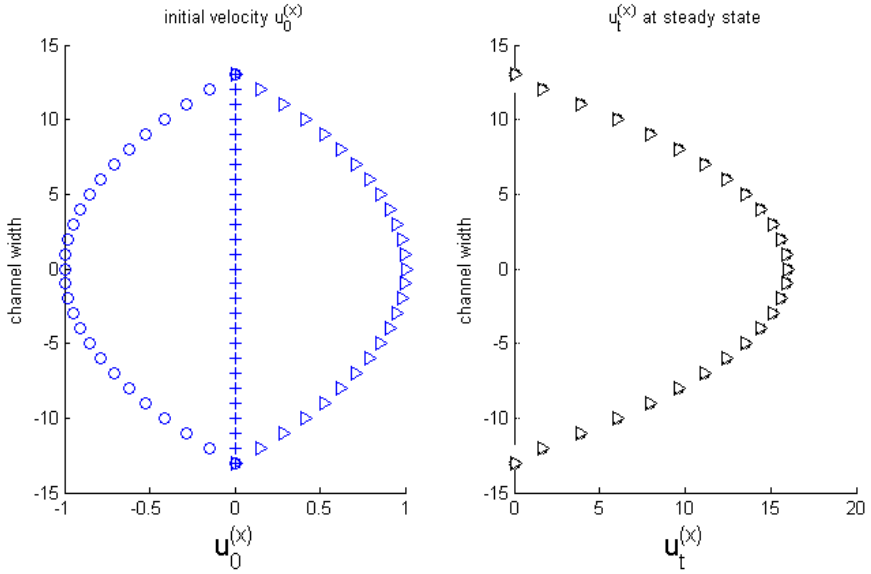

Figure 2.9: Illustration of the effect of varying initial velocities, fixing the kinematic viscosity and forcing

2.5.2 Flow in a Cavity

To model a flow in a cavity, we prescribe a no-slip boundary conditions on all 4 sides of the domain. Hence, in this setup, there is no inlet and outlet for the flow. We observe what happens to the flow when there is a nonzero initial velocity and zero forcing to drive the flow.

We run simulations for 25×30 lattice of fluid particles, kinematic viscosity of 1 (giving $\tau = 3.5$) and macroscopic density ρ_0 of 1 for every node. We assign initial $u_0^{(x)}$ to have a parabolic profile having a maximum value $u_{0,\max}^{(x)} = 1$ while $u_0^{(y)} = 0$ and volume force vector $\mathbf{F} = (0, 0)^T$ is applied. We observe the flow until it reaches the steady-state. Figure 2.10 shows the steady-state $u_t^{(x)}, u_t^{(y)}$ and p reached after 510 iterations. As expected, the velocity zeroes out (magnitude of 10^{-6}) throughout time and the pressure remains constant throughout the domain (approximately) 1.

Once again, we compare the effect of the varying the viscosity in reaching the steady-state when zero volume force is applied shown in Table 2.2. We set $u_0^{(x)}$ to have a parabolic profile with $u_{0,\max}^{(x)} = 1$ and $u_0^{(y)} = 0$. As expected, larger viscosity implies less iterations to reach the steady-state.

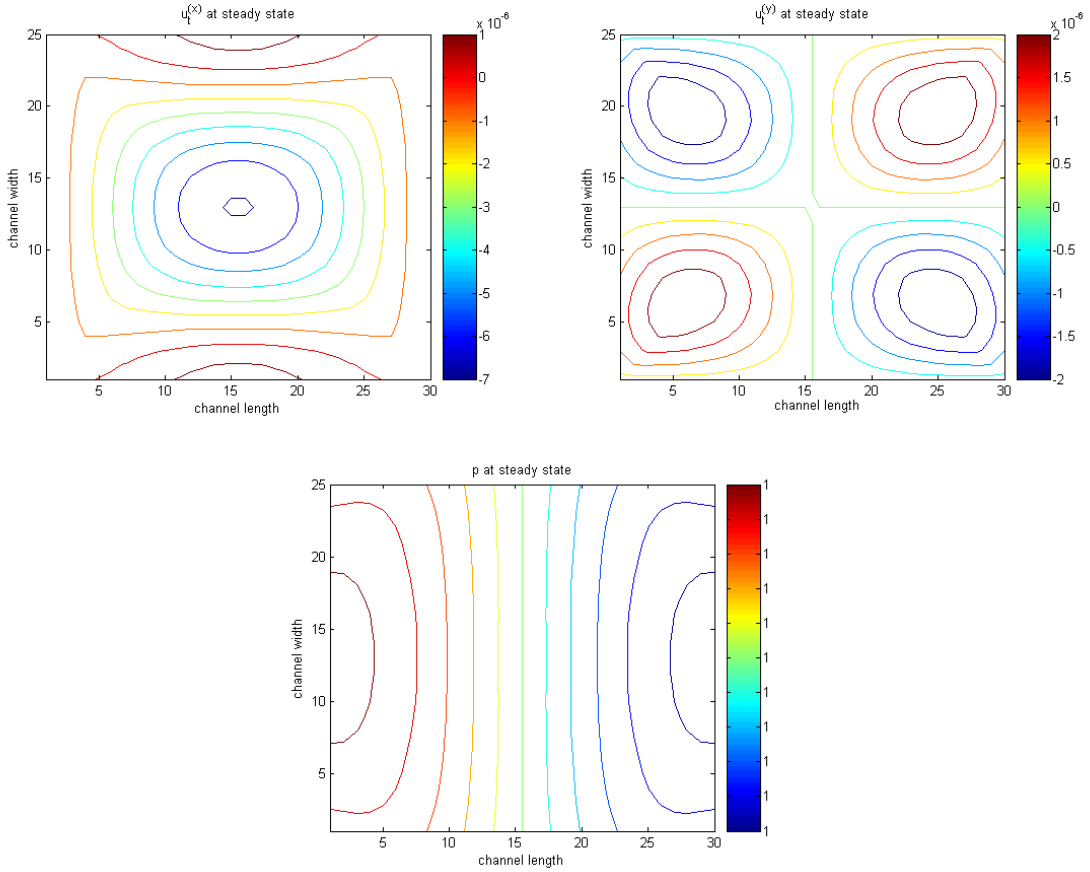

Figure 2.10: The macroscopic properties $u_t^{(x)}, u_t^{(y)}$ and p at the steady-state at $t = 510$

viscosity μ	τ	no. of iterations
0.75	2.75	667
1	3.5	510
1.5	5	350
2	6.5	252

Table 2.2: Comparison of the effect of varying the viscosity in reaching the steady-state flow in a cavity

Chapter 3

Optimization using the discrete adjoint method

Optimization is a familiar concept we encounter in our everyday lives as we are faced daily with decisions to make. In decision-making, one normally chooses what is best to serve his interest among the choices bounded by the limits of his resources. In this chapter, we shall be looking on a certain type of an optimization problem and examine ways of solving this. We shall attempt to formulate a way of solving the problem in such a way that the optimization can be done inexpensively and can be applied in the context of fluid simulation using the LBM.

3.1 Optimal control problems

Consider the problem

$$\min_{v \in V} \hat{\varphi}(v) \quad \text{subject to} \quad \hat{Q}(v) = 0 \tag{3.1}$$

where $\hat{\varphi} : V \to \mathbb{R}$ and $\hat{Q} : V \to Z$, V and Z are Banach spaces, and where $\hat{Q}(v) = 0$ typically represents a partial differential equation or a system of coupled partial differential equations (PDEs). In this optimization problem, the function $\hat{\varphi}$ is known as the **objective function** or **cost function** and $\hat{Q}(v) = 0$ is the **optimization constraint**. The domain V of $\hat{\varphi}$ is the search space while the elements $v \in V$ are the **feasible solutions**. A feasible solution that minimizes the objective function is called an **optimal solution**.

The optimization variable v admits a natural splitting into 2 parts: a **state** $w \in W$ and a **control** $\alpha \in Y$, where W and Y are Banach spaces. The problem (3.1) can now be

formulated as

$$\min_{(w,\alpha)\in W\times Y} \varphi(w,\alpha) \quad \text{subject to} \quad Q(w,\alpha)=0 \tag{3.2}$$

where $\varphi : V = W \times Y \to \mathbb{R}$ and $Q : V = W \times Y \to Z$, where W, Y and Z are function spaces. We call the constraint $Q(w,\alpha) = 0$ as the **state equation**.

The splitting of $v = (w,\alpha)$ is a typical strategy in optimization and problems of this structure are called **optimal control problems**. Here $w \in W$ describes the state of the considered system represented by $Q(w,\alpha) = 0$. Assuming the well-posedness of $Q(w,\alpha) = 0$, for every $\alpha \in Y$, there exist corresponding unique $w(\alpha)$. The control $\alpha \in Y$ is a parameter that we shall aim to determine such that α drives the system to optimality.

3.2 Strategies in solving optimal control problems

One of the possible ways of attacking an optimization problem is by using gradient-based methods. Here, the optimization procedure requires the first derivatives of the objective function. Difficulties and challenges arise, however, when an attempt to obtain these derivatives are carried out. As emphasized in [19], one consideration that needs to be taken into account is whether such derivative information is available, not only in the sense that it has to exist and be continuous, but also whether their values can be computed with reasonable expense.

In solving the optimization problem (3.2) using gradient-based methods, the principal step is determining the gradient of the objective function $\varphi(w,\alpha)$, i.e., evaluating the derivative of the function $\varphi(w,\alpha)$ with respect to parameter α in the situation where φ depends on α indirectly, via the intermediate variable $w(\alpha)$. There are number of ways of computing the gradient which can be categorized mainly into: 1. Sensitivity or Forward approach and 2. Adjoint or Backward or Reverse approach.

In the sensitivity approach, generally speaking, we compute the sensitivity of the solution (state) w of the state equation with respect to the control α in some way. The gradient of the objective function φ is then evaluated by applying the chain rule of differentiation, using the sensitivity $\dfrac{\partial w}{\partial \alpha}$ where we will ultimately have

$$\frac{D\varphi(w,\alpha)}{D\alpha} = \frac{\partial \varphi}{\partial w}\frac{\partial w}{\partial \alpha} + \frac{\partial \varphi}{\partial \alpha}.$$

Computing the matrix-vector product $\dfrac{\partial \varphi}{\partial w}\dfrac{\partial w}{\partial \alpha}$ can rather be expensive since this will consist of an entire state equation for each control.

On the other hand, using the adjoint approach, we omit the sensitivities of w and instead we introduce a variable ψ we call the adjoint state as shall be explained in details in Section 3.3, where ultimately we solve the equation

$$\frac{D\varphi(w,\alpha)}{D\alpha} = -\psi^T \frac{\partial Q}{\partial \alpha} + \frac{\partial \varphi}{\partial \alpha}. \tag{3.3}$$

In applying the adjoint method, the terms appearing in equation (3.3) need to be computed. Suppose we are given a system of PDE serving as the state equation, the optimal control problem constraint. However, instead of using this system of PDE in its continuous form, we can take instead a discretized representation of the system of PDEs (i.e. a set of algebraic equations and algebraic expressions as an approximation for the system of PDE) and w the discrete representation of the solution of the system. This discretized representation shall function as the constraint $Q(w,\alpha) = 0$. And from this shall follow the computation of the terms of the gradient equation. We call this method the **discrete adjoint method** and this whole procedure is coined in [10] as "**discretize then optimize**".

The description "discretize then optimize" distinguishes this to another way of tackling the optimization problem. This approach is done with the continuous $Q(w,\alpha) = 0$ as a starting point. From here, the system is differentiated at the continuous level to obtain the adjoint formulation of the problem defining an adjoint solution and then the adjoint discretization follows. This approach is called the **continuous adjoint method** since the adjoint technique is applied on the continuous level and the whole procedure is described as "**optimize then discretize**."

In this work, we shall be heading towards the "discretize then optimize" track and among the possible methods for a class of gradient-based optimization approaches, we shall put our interest in the application of the discrete adjoint method for the advantage it offers in computational efficiency.

3.3 The discrete adjoint method

In the following 2 subsections, we present 2 ways of deriving the so-called adjoint equation and gradient equation we use for the discrete adjoint method.

3.3.1 The duality viewpoint approach

As a motivation (also used in [15] and [6]), let us assume a matrix A and vectors g and c are known. Suppose we want to compute

$$g^T b \text{ such that } Ab = c. \tag{3.4}$$

The straightforward treatment to this problem is to solve b and then compute $g^T b$. Alternatively, as another treatment we introduce s and compute

$$s^T c \text{ such that } A^T s = g. \tag{3.5}$$

Observe that

$$s^T c = s^T (Ab) = (A^T s)^T b = g^T b.$$

Hence, solving $s^T c$ is accomplishing the original aim of solving $g^T b$. The constructed problem (3.5) is called the **dual problem** of (3.4). The advantage of resorting to solve the dual problem becomes apparent when we have an aggregate of vectors b_i and c_i, forming matrices B and C. Thus, (3.4) becomes

$$g^T B \text{ such that } AB = C$$

and the dual problem reformulates to

$$s^T C \text{ such that } A^T s = g.$$

Clearly, the dual is easier to solve in eventually computing the value of $g^T B$.

Now, recall our optimal control problem (3.2). Suppose in particular, that the state equation is an evolution equation that can be written as

$$Q_t(w_t, w_{t-1}, \alpha_{t-1}) = 0 \tag{3.6}$$

where the subscripts denote time indexing, i.e. w_i, α_i are the state and the control at time i, respectively. Assuming that w_0 is a given initial state, we then evolve into T subsequent states $w_1, \ldots, w_T$. We can write (3.6) in a form of an explicit update rule

$$w_t = Q_t(w_{t-1}, \alpha_{t-1}). \tag{3.7}$$

Thus, (3.2) reformulates into

$$\min_{w_t \in \mathbb{R}^m, \alpha_t \in \mathbb{R}^n} \varphi(w_t, \alpha_t) \quad \text{subject to} \quad w_t = Q_t(w_{t-1}, \alpha_{t-1}), \quad t = 1, \ldots, T. \tag{3.8}$$

The reason for this assumption on the form of the state equation will become apparent in the following section.

Let us collect the states and control to form vectors $W = [w_1, \ldots, w_T]^T$ and $\alpha = [\alpha_1, \ldots, \alpha_T]^T$ and define a vector function

$$Q(W, \alpha) = [Q_1(w_0, \alpha_0), \ldots, Q_T(w_{T-1}, \alpha_{T-1})]^T,$$

and therefore we can write our optimization constraint (3.7) as

$$W = Q(W, \alpha). \tag{3.9}$$

Now computing the gradient of φ with respect to control α,

$$\frac{d\varphi}{d\alpha} = \frac{\partial\varphi}{\partial W}\frac{dW}{d\alpha} + \frac{\partial\varphi}{\partial\alpha}. \tag{3.10}$$

Computing this directly is extremely costly, as the matrix $\dfrac{dW}{d\alpha}$ consists of an entire state sequence for each control. As we shall see, the adjoint method provides a way of side-stepping this computation while still arriving at exact gradient of φ [15]. Differentiating the constraint equation (3.9) gives us a linear constraint on the derivative matrix $\dfrac{dW}{d\alpha}$. Thus, the first term of equation (3.10) calls for calculating

$$\frac{\partial\varphi}{\partial W}\frac{dW}{d\alpha} \quad \text{such that} \quad \left(I - \frac{\partial Q}{\partial W}\right)\frac{dW}{d\alpha} = \frac{\partial Q}{\partial\alpha}. \tag{3.11}$$

As we have seen in the movitation demonstrated above, using the dual, (3.11) can be efficiently computed as

$$P^T\frac{\partial Q}{\partial\alpha} \quad \text{such that} \quad \left(I - \frac{\partial Q}{\partial W}\right)^T P = \frac{\partial\varphi}{\partial W}^T. \tag{3.12}$$

To calculate P, let us rewrite the constraint in equation (3.12) as

$$P = \left(\frac{\partial Q}{\partial W}\right)^T P + \frac{\partial\varphi}{\partial W}^T. \tag{3.13}$$

As W is an aggregate of a sequence of forward states $w_1, \ldots, w_T$, similarly, we may view P as an aggregate of a sequence of variables $\psi_1, \ldots, \psi_T$ we call the **adjoint states**. (3.13) implies $\psi_T = \dfrac{d\varphi}{dw_T}^T$ and

$$\psi_t = \left(\frac{\partial Q_t}{\partial w_t}\right)^T \psi_{t+1} + \frac{\partial\varphi}{\partial w_t}^T, \quad t = T-1, \ldots, 1. \tag{3.14}$$

Finally, the gradient in (3.10) is computed by

$$\frac{d\varphi}{d\alpha} = P^T \frac{\partial Q}{\partial \alpha} + \frac{\partial \varphi}{\partial \alpha},$$

which is equivalent to

$$\frac{d\varphi}{d\alpha_t} = \psi_t^T \frac{\partial Q}{\partial \alpha_t} + \frac{\partial \varphi}{\partial \alpha_t}, \qquad t = 1, \ldots, T. \tag{3.15}$$

(3.14) is called the **adjoint equation** and (3.15) is called the **gradient equation** and the procedure of obtaining these equations using the dual problem and introducing the adjoint variables is referred to in literature as the **duality viewpoint approach**.

3.3.2 The Lagrange viewpoint approach

The adjoint equation (3.14) and the gradient equation (3.15) can be derived in another manner often referred to in literature as the **Lagrange viewpoint approach**. Once again, let us recall the reformulated optimal control problem (3.8) and suppose the objective function φ has the form

$$\varphi = \sum_{t=1}^{T} \varphi_t(w_t, \alpha_t). \tag{3.16}$$

Let $g_t \in \mathbb{R}^{n\times 1}$ be the gradient of φ_t with respect to control $\alpha_t \in \mathbb{R}^{n\times 1}$.

As outlined in [10], to solve g_t, let us use $u_t = d\alpha_t \in \mathbb{R}^{n\times 1}$ and $z_t = dw_t \in \mathbb{R}^{m\times 1}$ and differentiate the state equation $w_t = Q_t(w_{t-1}, \alpha_{t-1})$. We have

$$\underbrace{z_t}_{\in \mathbb{R}^{m\times 1}} = \underbrace{\frac{\partial Q_t}{\partial w_{t-1}}(w_{t-1}, \alpha_{t-1})}_{\in \mathbb{R}^{m\times m}} \underbrace{z_{t-1}}_{\in \mathbb{R}^{m\times 1}} + \underbrace{\frac{\partial Q_t}{\partial \alpha_{t-1}}(w_{t-1}, \alpha_{t-1})}_{\in \mathbb{R}^{m\times n}} \underbrace{u_t}_{\in \mathbb{R}^{n\times 1}}, \qquad z_0 = 0. \tag{3.17}$$

Differentiating (3.16),

$$\underbrace{d\varphi}_{\in \mathbb{R}} = \sum_{t=1}^{T} \left[\underbrace{\nabla_w \varphi_t(w_t, u_t)}_{\in \mathbb{R}^{m\times 1}} \cdot z_t \right] + \sum_{t=1}^{T} \left[\underbrace{\nabla_\alpha \varphi_t(w_t, u_t)}_{\in \mathbb{R}^{n\times 1}} \cdot u_t \right], \tag{3.18}$$

where $\nabla_{w_t} \varphi_t(w_t, u_t)$ is the gradient of φ_t with respect to the state variables and $\nabla_{\alpha_t} \varphi_t(w_t, u_t)$ is the gradient of φ_t with respect to the control variables.

Setting

$$G_t = \frac{\partial Q_t}{\partial w_{t-1}}(w_{t-1}, \alpha_{t-1}) \qquad H_t = \frac{\partial Q_t}{\partial \alpha_{t-1}}(w_{t-1}, \alpha_{t-1})$$

$$\gamma_t = \nabla_{w_t} \varphi_t(w_t, u_t) \qquad h_t = \nabla_{\alpha_t} \varphi_t(w_t, u_t),$$

(3.17) becomes

$$z_t = G_t z_{t-1} + H_t u_t. \tag{3.19}$$

Multiplying $\psi_t \in \mathbb{R}^{m\times 1}$ to (3.19), we get

$$\underbrace{\psi_t^T}_{\mathbb{R}^{1\times m}}\ \underbrace{z_t}_{\mathbb{R}^{m\times 1}} = \underbrace{\psi_t^T}_{\mathbb{R}^{1\times m}}\ \underbrace{G_t}_{\mathbb{R}^{m\times m}}\ \underbrace{z_{t-1}}_{\mathbb{R}^{m\times 1}} + \underbrace{\psi_t^T}_{\mathbb{R}^{1\times m}}\ \underbrace{H_t}_{\mathbb{R}^{m\times n}}\ \underbrace{u_t}_{\mathbb{R}^{n\times 1}}$$

$$\psi_t \cdot z_t = G_t^T \psi_t \cdot z_{t-1} \ + \ H_t^T \psi_t \cdot u_t$$

and summing up through time, we get

$$\sum_{t=1}^{T} \psi_t \cdot z_t = \sum_{t=1}^{T} G_t^T \psi_t \cdot z_{t-1} \ + \ \sum_{t=1}^{T} H_t^T \psi_t \cdot u_t.$$

Since $z_0 = 0$, we reduce to

$$\begin{aligned}
\sum_{t=1}^{T} \psi_t \cdot z_t &= \sum_{t=2}^{T} G_t^T \psi_t \cdot z_{t-1} \ + \ \sum_{t=1}^{T} H_t^T \psi_t \cdot u_t \\
\sum_{t=1}^{T} \psi_t \cdot z_t &= \sum_{t=1}^{T-1} G_{t+1}^T \psi_{t+1} \cdot z_t \ + \ \sum_{t=1}^{T} H_t^T \psi_t \cdot u_t \\
0 &= -\psi_T \cdot z_T - \sum_{t=1}^{T-1} \psi_t \cdot z_t + \sum_{t=1}^{T-1} G_{t+1}^T \psi_{t+1} \cdot z_t \ + \ \sum_{t=1}^{T} H_t^T \psi_t \cdot u_t \\
0 &= -\psi_T \cdot z_T + \sum_{t=1}^{T-1} \left(-\psi_t + G_{t+1}^T \psi_{t+1}\right) \cdot z_t \ + \ \sum_{t=1}^{T} H_t^T \psi_t \cdot u_t.
\end{aligned} \tag{3.20}$$

On the other hand, (3.18) becomes

$$\begin{aligned}
d\varphi &= \sum_{t=1}^{T} \gamma_t \cdot z_t \ + \ \sum_{t=1}^{T} h_t \cdot u_t \\
d\varphi &= \gamma_T \cdot z_T \ + \ \sum_{t=1}^{T-1} \gamma_t \cdot z_t \ + \ \sum_{t=1}^{T} h_t \cdot u_t
\end{aligned} \tag{3.21}$$

Adding (3.20) and (3.21), we obtain

$$d\varphi = (-\psi_T + \gamma_T) \cdot z_T + \sum_{t=1}^{T-1} \left(-\psi_t + \gamma_t + G_{t+1}^T \psi_{t+1}\right) \cdot z_t \ + \ \sum_{t=1}^{T} \left(H_t^T \psi_t + h_t\right) \cdot u_t. \tag{3.22}$$

In choosing ψ_T so that the term with z_t in (3.22) vanishes, we then require

$$\psi_T = \gamma_T, \quad \psi_t = G_{t+1}^T \psi_{t+1} + \gamma_t, \quad t = T-1, \ldots, 1. \tag{3.23}$$

(3.23) is called the adjoint equation, which solves states ψ_t, called the adjoint states, backward in time. Fulfilling the adjoint equation, we then have

$$\begin{aligned} d\varphi &= \sum_{t=1}^{T} \left(H_t^T \psi_t + h_t\right) \cdot u_t \\ d\varphi_t &= \left(H_t^T \psi_t + h_t\right) \cdot u_t, \qquad t = T-1, \ldots, 1. \end{aligned}$$

Therefore, the gradient $g_t \in \mathbb{R}^n$ of φ_t with respect to control $\alpha \in \mathbb{R}^n$ is given by the gradient equation

$$g_t = \frac{d\varphi_t}{d\alpha_t} = H_t^T \psi_t + h_t, \qquad t = 1, \ldots, T. \tag{3.24}$$

In this framework, adjoint variables, i.e. the multiplied variable ψ_t and chosen such that the term with z_t in (3.22) vanishes, are viewed as Lagrange multipliers (see [6].) Notice that the adjoint equation (3.23) and gradient equation (3.24) are the same as (3.14) and (3.15) derived in the previous subsection using duality.

As we observe in both approaches, the adjoint method is a process of establishing the gradient via a set of introduced variables we know as the adjoint states. This is done by simulating and collecting each states forward in time through an evolutionary state equation-operator and then passing through each corresponding adjoint equation-operator in reverse order utilizing the collected states to compute the adjoint states as illustrated in Figure 3.3. This is the reason why the adjoint method is also called the backward or reverse method.

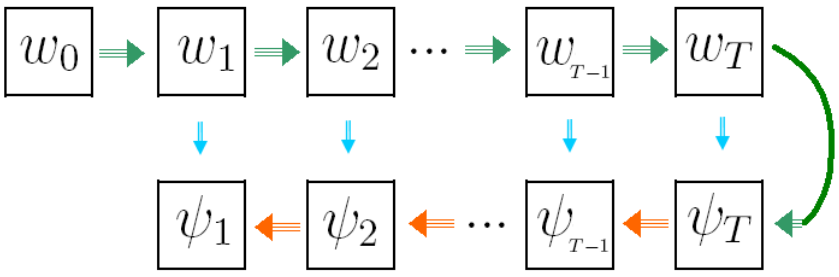

Figure 3.1: The states computed and collected forward in time and used later for computing the adjoint states backward in time. [15]

3.4 Optimal control of fluids based on LBM discretizations

Now that we are equipped with the method of determining the gradient of an objective function of an optimization problem, we shall now be placing what we gathered in the context of optimization involving fluid flows.

The optimal control problem (3.8) is recalled by

$$\min_{w_t \in \mathbb{R}^m, \alpha_t \in \mathbb{R}^n} \varphi(w_t, \alpha_t) \quad \text{subject to} \quad w_t = Q_t(w_{t-1}, \alpha_{t-1}), \quad t = 1, \dots, T,$$

the optimization problem we considered where the state equation takes the form of explicit update rule involving the states. Consider some particular 2-dimensional flow as a reference which is characterized by its macroscopic properties $\rho, u^{(x)}, u^{(y)}, p$ we regard as the **reference** or **desires states**. We create an objective function φ which essentially gives an indication of how closely a simulation mimics this reference flow. This quantity we intend to optimize is given by

$$\varphi(w_t, \alpha_t) = \frac{1}{2} \int_{\Omega \times \mathcal{T}} \left| w_t - w_t^d \right|^2 + \frac{\lambda}{2} \int_{\Omega \times \mathcal{T}} |\alpha_t|^2 , \tag{3.25}$$

where the state variable w_t at time t consists of macroscopic properties at time t of the simulated flow, say $w_t = (\rho_t, u_t^{(x)}, u_t^{(y)}, p_t)^T$, the control variable α_t at time t consists of the external volume force, say $\alpha_t = (\alpha_t^{(x)}, \alpha_t^{(y)})^T$ and w_t^d gives the reference or desired state. The first term in the objective function measures the distance of w_t and w_t^d and the second term is a regularization term with parameter $\lambda > 0$. In practice, typically $\lambda \in [10^{-5}, 10^{-3}]$ [10].

At this point, it is clear that a fluid flow phenomena is described by the Navier-Stokes equation and is characterized by the macroscopic fluid properties. We optimize the objective function such that the states satisfies, hence, are subject to, the Navier-Stokes equation acting as the constraint. However, as we have already mentioned in Section 3.2, the direction "discretize then optimize" we are heading to concerns a state equation that is a discretized representation of the constraint as the starting point. This is where the LBM comes into play. The LBM solver shall portray the role of $w_t = Q_t(w_{t-1}, \alpha_{t-1})$, computing the states w_t forward in time functioning as the discretized representation of the Navier-Stokes equation. A schematic diagram for our optimal control problem is given in Figure 3.2.

From this idea we have established follows the application of the discrete adjoint method. We note that we aim to evaluate the derivative of objective function φ with respect to control α, in this situation where φ depends on α indirectly, via the state w as an intermediate variable as discussed in Section 3.2, which involves the differentiation of the

objective function leading to the gradient equation and the computation of the terms present in the gradient equation.

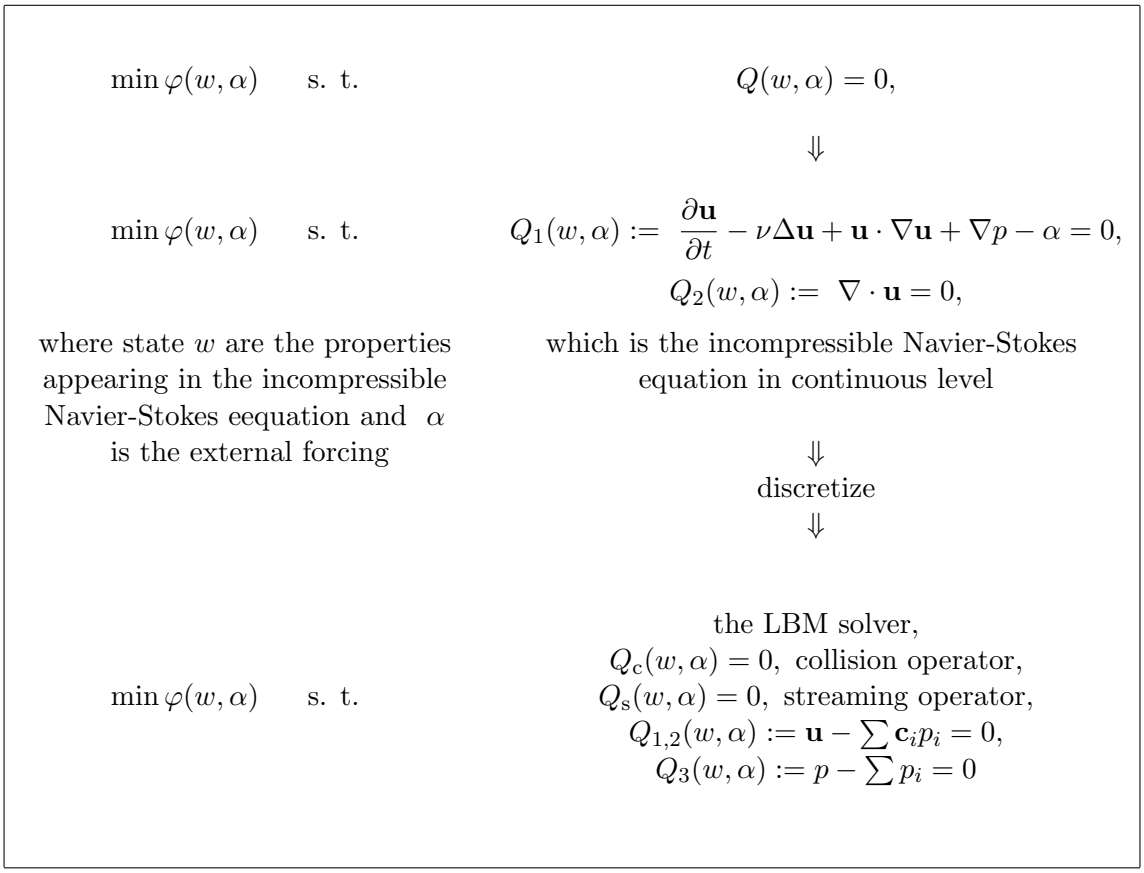

Figure 3.2: A schematic outline of the "discretize then optimize" approach for optimization using the incompressible Navier-Stokes equation as the constraint

3.4.1 Computations of the adjoint states and gradient

We follow in our computation the derivation of the gradient equation (3.24)

$$g_t = \frac{d\varphi_t}{d\alpha_t} = H_t^T \psi_t + h_t, \quad t = 1, \ldots, T.$$

in Subsection 3.3.2 where ψ_t is obtained from the adjoint equation (3.23)

$$\psi_T = \gamma_T, \quad \psi_t = G_{t+1}^T \psi_{t+1} + \gamma_t, \quad t = T-1, \ldots, 1,$$

In our implementation, let us set the state w_t to be comprised of the macroscopic properties explicitly appearing in the incompressible Navier-Stokes equation, which are also used in the LBM model of [8] for the incompressible Navier-Stokes equation, i.e. $w_t = (u_t^{(x)}, u_t^{(y)}, p_t)^T$ and set the control variable to be the 2-dimensional external volume force $\alpha_t = (\alpha_t^{(x)}, \alpha_t^{(y)})^T$.

Writing $\varphi = \sum_{t=1}^{T} \varphi_t(w_t, \alpha_t)$, then from (3.25) we have

$$\varphi(w_t, \alpha_t) = \sum_{t=1}^{T} \frac{1}{T-1} \left(\frac{1}{2} \int_\Omega \left| w_t - w_t^d \right|^2 + \frac{\lambda}{2} \int_\Omega |\alpha_t|^2 \right),$$

hence,

$$\varphi_t(w_t, \alpha_t) = \frac{1}{T-1} \left(\frac{1}{2} \int_\Omega \left| w_t - w_t^d \right|^2 + \frac{\lambda}{2} \int_\Omega |\alpha_t|^2 \right).$$

We need the derivatives

$$\gamma_t = \nabla_{w_t} \varphi_t(w_t, y_t), \qquad h_t = \nabla_{\alpha_t} \varphi_t(w_t, y_t),$$

where $y_t = d\alpha_t$. This is given by

$$\begin{aligned} \frac{\partial \varphi_t}{\partial w_t}(w_t, \alpha_t) &= \frac{1}{T-1} \left(w_t - w_t^d \right) = \gamma_t \in \mathbb{R}^{3\times 1} \\ \frac{\partial \varphi_t}{\partial \alpha_t}(w_t, \alpha_t) &= \frac{1}{T-1} \left(\lambda \alpha_t \right) = h_t \in \mathbb{R}^{2\times 1} \end{aligned}$$

Next we need, $G_t = \dfrac{\partial Q_t}{\partial w_{t-1}}(w_{t-1}, \alpha_{t-1})$ and $H_t = \dfrac{\partial Q_t}{\partial \alpha_{t-1}}(w_{t-1}, \alpha_{t-1})$. We follow the model of [8] for the incompressible Navier Stokes equation thus giving us the equations for the macroscopic properties, the states $u_t^{(x)}, u_t^{(y)}, p_t$ and time t as given by

$$\begin{aligned} u^{(x)}((x_i, y_j), t) &= p_1((x_i, y_j), t) - p_3((x_i, y_j), t) + p_5((x_i, y_j), t) \\ &\quad - p_6((x_i, y_j), t) - p_7((x_i, y_j), t) + p_8((x_i, y_j), t) \\ u^{(y)}((x_i, y_j), t) &= p_2((x_i, y_j), t) - p_4((x_i, y_j), t) + p_5((x_i, y_j), t) \\ &\quad + p_6((x_i, y_j), t) - p_7((x_i, y_j), t) - p_8((x_i, y_j), t) \\ p((x_i, y_j), t) &= \sum_{\hat{i}=1}^{9} p_{\hat{i}}((x_i, y_j), t). \end{aligned} \tag{3.26}$$

These properties in (3.26) at time t are functions of the pressure distributions $p_{\hat{i},t}$ at time t. Now, the pressure distribution $p_{\hat{i},t}$ at time t is a function of the collision term (that includes forcing) at time $t-1$, and further, the collision term at time $t-1$ is a function of the properties $u^{(x)}_{t-1},\ u^{(y)}_{t-1},\ p_{t-1}$ at time $t-1$. Upon tracing this chain of dependence we clearly observe the state at time t is an explicit function of the state at time $t-1$. The equations involved are broken down into full details in (A.1)-(A.4) in the Appendix.

To compute G_t,

$$w_t = Q_t(w_{t-1}, \alpha_{t-1})$$

can be written as

$$w_t = \begin{bmatrix} u^{(x)}_t \\ u^{(y)}_t \\ p_t \end{bmatrix} = \begin{bmatrix} Q^1_t(w_{t-1}, \alpha_{t-1}) \\ Q^2_t(w_{t-1}, \alpha_{t-1}) \\ Q^3_t(w_{t-1}, \alpha_{t-1}) \end{bmatrix}, \tag{3.27}$$

and thus,

$$G_t = \frac{\partial Q_t}{\partial w_{t-1}} = \begin{bmatrix} \frac{\partial Q^1_t}{\partial u^{(x)}_{t-1}} & \frac{\partial Q^1_t}{\partial u^{(y)}_{t-1}} & \frac{\partial Q^1_t}{\partial p_{t-1}} \\ \frac{\partial Q^2_t}{\partial u^{(x)}_{t-1}} & \frac{\partial Q^2_t}{\partial u^{(y)}_{t-1}} & \frac{\partial Q^2_t}{\partial p_{t-1}} \\ \frac{\partial Q^3_t}{\partial u^{(x)}_{t-1}} & \frac{\partial Q^3_t}{\partial u^{(y)}_{t-1}} & \frac{\partial Q^3_t}{\partial p_{t-1}} \end{bmatrix}.$$

Next, we compute the entries of G_t. Let us take for example $\dfrac{\partial Q^1_t}{\partial u^{(x)}_{t-1}}$, the $(1,1)^{\text{th}}$ entry of G_t. On the first equation of (3.26), we apply the differential operator $\dfrac{\partial}{\partial u^{(x)}_{t-1}}$, thus we have

$$\begin{aligned} \frac{\partial}{\partial u^{(x)}_{t-1}} u^{(x)}((x_i, y_j), t) \;=\; & \frac{\partial}{\partial u^{(x)}_{t-1}} p_1((x_i, y_j), t) - \frac{\partial}{\partial u^{(x)}_{t-1}} p_3((x_i, y_j), t) + \frac{\partial}{\partial u^{(x)}_{t-1}} p_5((x_i, y_j), t) \\ & - \frac{\partial}{\partial u^{(x)}_{t-1}} p_6((x_i, y_j), t) - \frac{\partial}{\partial u^{(x)}_{t-1}} p_7((x_i, y_j), t) + \frac{\partial}{\partial u^{(x)}_{t-1}} p_8((x_i, y_j), t). \end{aligned} \tag{3.28}$$

For each terms of (3.28), we have applied the differential operator $\dfrac{\partial}{\partial u_{t-1}^{(x)}}$ which means applying the chain rule tracing along the sequence of dependence that ultimately leads to $u_{t-1}^{(x)}$. This is demonstrated in full details in (A.7)-(A.10) of the Appendix. Similarly, the same procedure is applied to obtain the rest of the entries of G_t and once all the entries of G_t are computed, we proceed to the computation of the adjoint states from the adjoint equation (3.24)

$$\psi_T = \gamma_T, \quad \psi_t = G_{t+1}^T \psi_{t+1} + \gamma_t, \quad t = T-1, \ldots, 1.$$

From our computations, it is an observation worthy of emphasis that for each time t, the computation of the adjoint states through the adjoint equation done in reverse time can also be viewed as some sort of LBM since the procedure keeps the collision, forcing, streaming and application of rules for boundaries which operates on derivatives of the properties $u_t^{(x)}$, $u_t^{(y)}$, p_t and of the pressure distributions (or particle distributions, due to original LBM formulation.)

To visualize, Figure 3.3 shows the adjoint states of a simulated flow. First, the figures on the left column show the evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 200$ for fluid of viscosity 1, with initial velocity $u_0^{(x)}$ having a parabolic profile with maximum value $u_{0,\max}^{(x)} = 1$ and $u_0^{(y)} = 0$, driven by constant volume force $\alpha = (-1, 0)$. The trend of $u_t^{(x)}$ can be followed as the parabola marked by $\times$ represents $u_1^{(x)}$, the parabola marked by $\circ$ gives $u_{10}^{(x)}$, $\triangleright$ gives $u_{50}^{(x)}$, $*$ gives $u_{100}^{(x)}$, $\cdot$ is for $u_{180}^{(x)}$ and $+$ is for $u_{200}^{(x)}$. $u_t^{(x)}$ is characterized by computational noise and p remains 1 for each node throughout time $t = 1$. We assign the desired states to be those driven by constant force $\alpha = (0.1, 0)$ from time $t = 1, \ldots, 200$. Computed through the adjoint equation (3.23), we see the adjoint states $\psi_t^{(1)}, \psi_t^{(2)}$ and $\psi_t^{(3)}$ on the left column of Figure 3.3. It is interesting to observe the trends of $\psi_t^{(1)}$ and $\psi_t^{(2)}$, reverse in time, followed as the markings: $+$ for $t = 200$, $\cdot$ for $t = 180$, $*$ gives $t = 100$, $\triangleright$ gives $t = 50$, $\circ$ gives $t = 10$ and $\times$ represents $t = 1$. From $t = 200$ to $t = 1$, $\psi_t^{(1)}$ grows and extends along the negative direction until it reaches a peak and retracts back to zero. $\psi_t^{(2)}$ grows and extends along the negative and positive direction until it reaches a peak and retracts back to zero and $\psi_t^{(3)}$ is characterized by numerical noise. As an interpretation, by looking at the construction of the adjoint equation, $\psi_t^{(1)}$ can be viewed as a combination of the rate of changes of $u^{(x)}, u^{(y)}$ and p with respect to $u^{(x)}$. $\psi_t^{(2)}$ can be regarded as a combination of the rate of changes of $u^{(x)}, u^{(y)}$ and p with respect to $u^{(y)}$, and similarly, $\psi_t^{(3)}$, with respect to p.

Similar to the computation of G_t, we determine $H_t = \dfrac{\partial Q_t}{\partial \alpha_{t-1}}(w_{t-1}, \alpha_{t-1})$. Following the

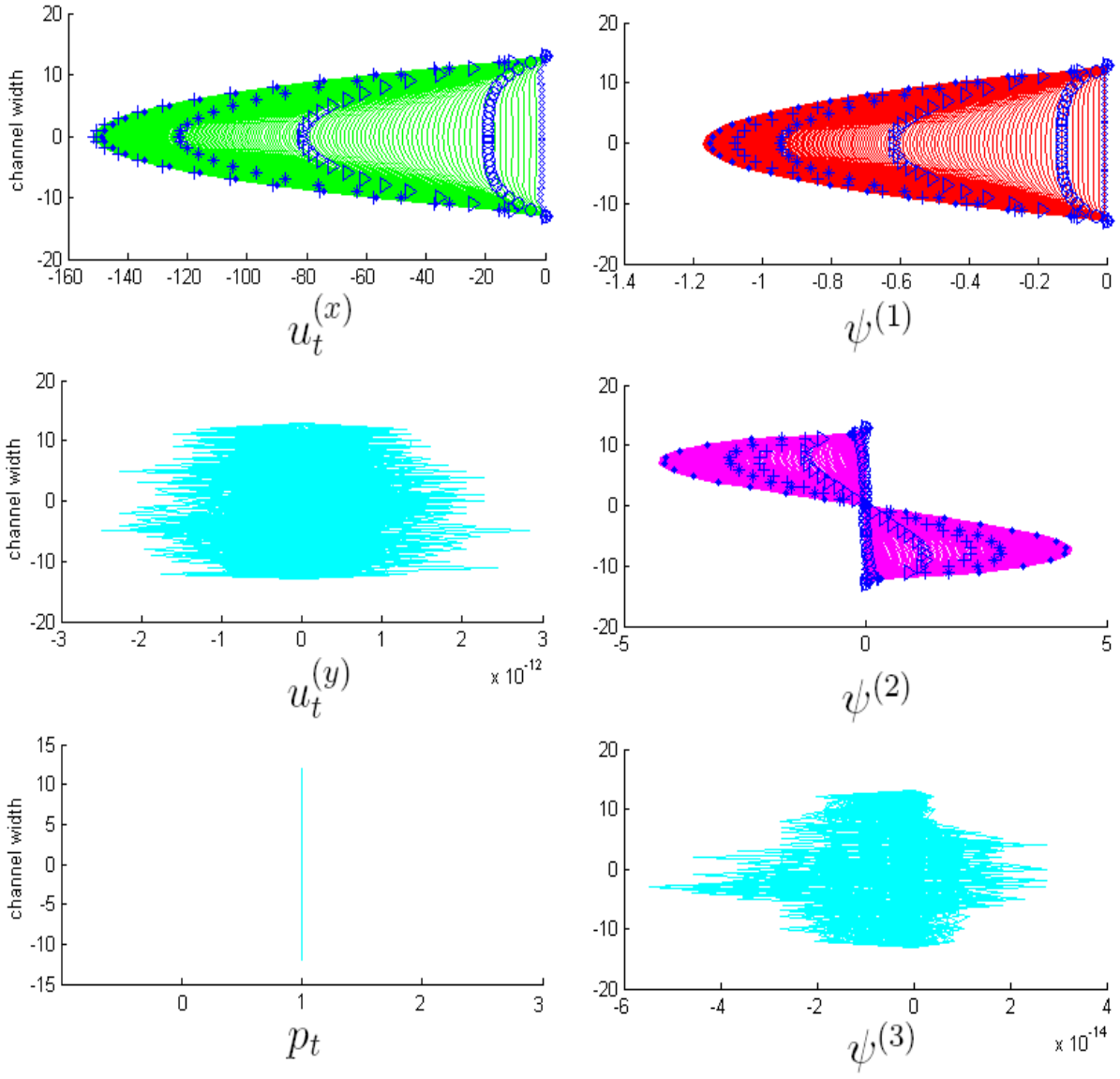

Figure 3.3: The evolution of the macroscopic properties $u^{(x)}, u^{(y)}$ and p from time $t = 1, \ldots, 200$, and the corresponding adjoint states computed from time $t = 200, \ldots, 1$

breakdown of Q_t given by (3.27), we have

$$H_t = \frac{\partial Q_t}{\partial \alpha_{t-1}} = \begin{bmatrix} \dfrac{\partial Q_t^1}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^1}{\partial \alpha_{t-1}^{(y)}} \\ \dfrac{\partial Q_t^2}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^2}{\partial \alpha_{t-1}^{(y)}} \\ \dfrac{\partial Q_t^3}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^3}{\partial \alpha_{t-1}^{(y)}} \end{bmatrix},$$

whose entries are also obtained by applying the differential operator $\dfrac{\partial}{\partial \alpha_{t-1}^{(x)}}$ or $\dfrac{\partial}{\partial \alpha_{t-1}^{(y)}}$ which means applying the chain rule along the sequence of dependence that ultimately leads to $\alpha_{t-1}^{(x)}$ or $\alpha_{t-1}^{(y)}$ demonstrated in full details in (A.13)-(A.16) of the Appendix. Finally, the gradient is computed through

$$g_t = H_t^T \psi_t + h_t, \quad t = 1, \ldots, T.$$

As a side remark, we may consider when the gradient is computed with the finite difference approximation

$$\nabla \varphi(\alpha) \cdot \mathbf{e}_i \approx \frac{\varphi(\alpha + h\mathbf{e}_i) - \varphi(\alpha)}{h},$$

where $\mathbf{e}_i$ is a vector with 1 on the i^{th} position and 0 elsewhere, as a simple test and compare this with the results of the discrete adjoint method we discussed above. To illustrate, assuming a constant volume force $\alpha = (\alpha^{(x)}, \alpha^{(y)})$ driving the flow acting as a control, let us set the desired states to be those driven by constant force $\alpha = (0.1, 0)$, set $h = 10^{-7}$ and compute the gradient of the objective function with respect to the control using discrete adjoint method and the finite difference approximation given above. Figure 3.4 compares $\alpha^{(x)}$-component of the gradient (i.e. the derivative of the objective with respect to $\alpha^{(x)}$) resulting from the finite difference approximation and the discrete adjoint method over samples of $\alpha \in [-0.6, 0.9] \times [-0.05, 0.05]$ as the bottom figure shows their difference that is within 10^{-6} order of magnitude. Figure 3.5 compares the $\alpha^{(y)}$-component of the gradient for the two methods and the bottom figure shows their difference.

Note that in our discrete adjoint method implementation, for a certain control all the components of the gradient is computed within one forward sweep and one backward sweep as described in Section 3.3. In contrast, with the finite difference approximation, we have to do one forward simulation with control $\alpha + h\mathbf{e}_i$ to obtain the macroscopic quantities then solve for the value of the objective $\varphi(\alpha + h\mathbf{e}_i)$, and then perform another forward sweep for $\varphi(\alpha)$. And since we compute each component of the gradient, we

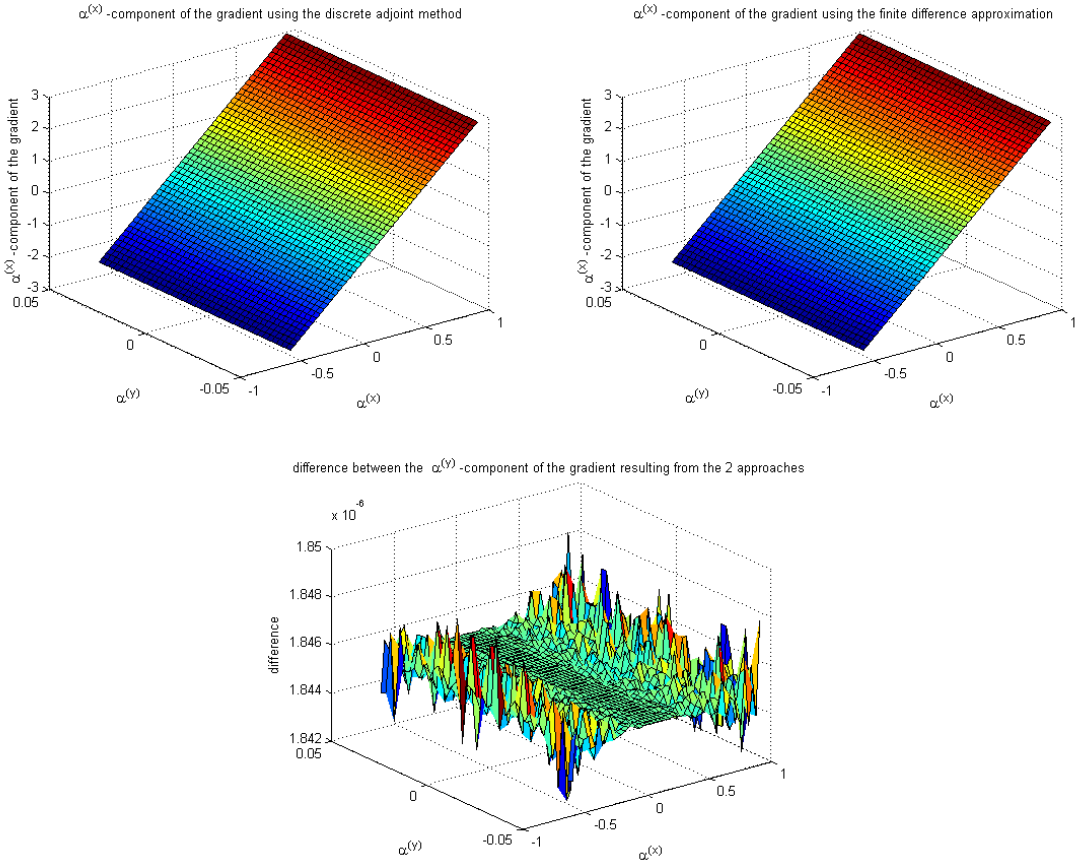

Figure 3.4: Comparing the $\alpha^{(x)}$-component of the gradient using the discrete adjoint method and the finite difference approximation

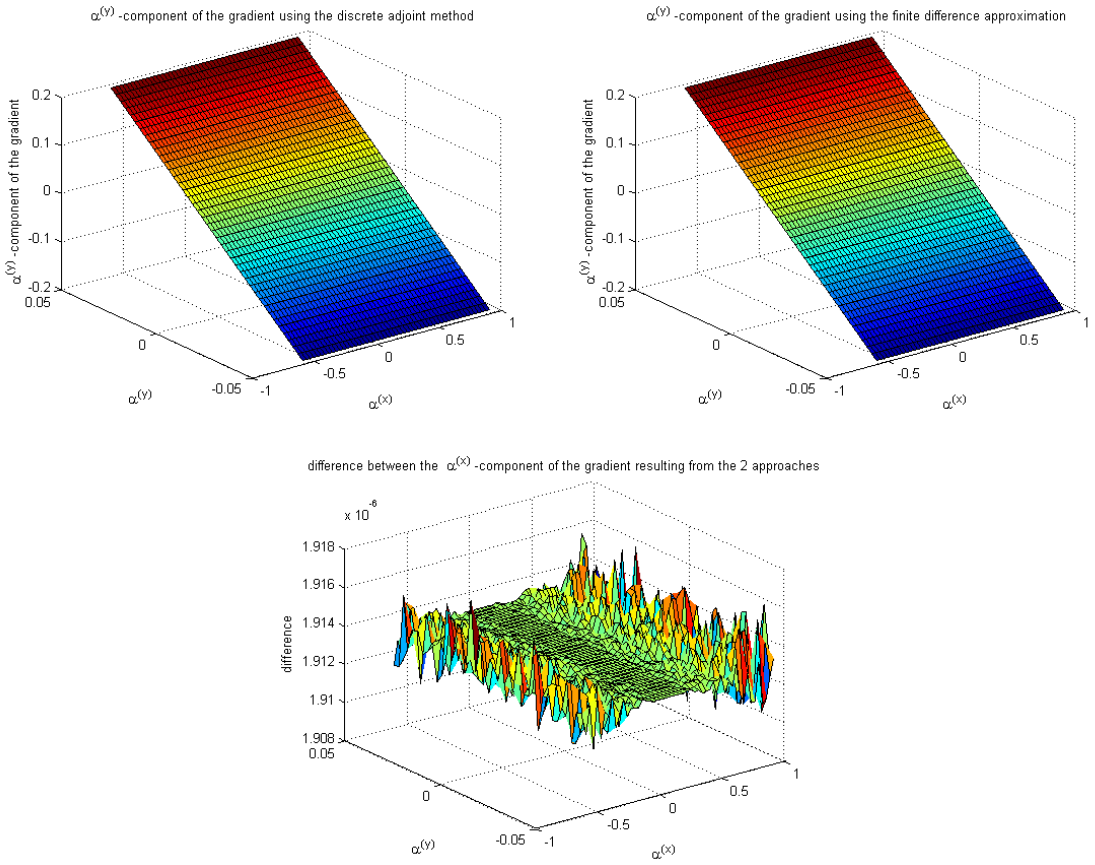

Figure 3.5: Comparing the $\alpha^{(y)}$-component of the gradient using the discrete adjoint method and the finite difference approximation

repeat the procedure for all i. Hence in general cases, finite difference approximation will be expensive when the control has many components (e.g. when other parameters are introduced in the control or when control varies for each node and for each point in time).

Chapter 4

Method of steepest descent

At this point it is clear that we seek for the control α that minimizes the objective φ. We wish to do this in a systematic way such that we may not be forced to compute the value of φ for every possible control α there is. We take note that computation of φ amounts to some expense and we do not want compute this value whenever they are not necessary. This choice of such systematic way shall then determine whether the optimal control problem is solved quickly or slowly or the optimal control is not found at all. However, since we do not wish to compute the value of φ for too many α's, we can not have a good overall picture of the shape of φ and we hope that φ does not take a sharp drop in some regions where we do not sample the value of φ.

4.1 Recognizing solutions

A natural question that arises is how to recognize or verify that indeed a variable α is a minimizer of a certain function φ. We follow for the discussion the definitions and theorems from [16].

Definition A point α^* is a **global minimizer** if $\varphi(\alpha^*) \leq \varphi(\alpha^*)$ for all $\alpha \in \mathcal{D}$, where $\mathcal{D}$ is the domain of interest.

The global minimizer is difficult to find because usually the knowledge of φ is only local. We have the following definition:

Definition A point α^* is a **local minimizer** if there is neighborhood $\mathcal{N}$ of α^* such that $\varphi(\alpha^*) \leq \varphi(\alpha^*)$ for all $\alpha \in \mathcal{N} \subset \mathcal{D}$.

Most algorithms are able to find only a local minimizer, which is the point that achieves the smallest value of φ in its neighborhood. It is usually difficult to find the global minimizers for functions that are wavy (with many oscillations) as algorithms for finding the global minimizers get caught up and stuck at local minimizers. In contrast, when functions are smooth, there are more efficient and practical ways to identify local minima. Further, an important set of special of functions known as convex functions are functions for which every local minimizer is also a global minimizer. To study the minimizers of smooth functions we begin with the following theorem:

Theorem 4.1.1 *(Taylor's Theorem) Suppose that $\varphi : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $s \in \mathbb{R}^n$. Then we have that*

$$\varphi(\alpha + s) = \varphi(\alpha) + \nabla\varphi(\alpha + \xi s)^T s$$

for some $\xi \in (0, 1)$.

Assuming α^* is a local minimizer, we try to determine implications of this on $\nabla\varphi(\alpha^*)$. We shall call these the necessary conditions for optimality. We have the following theorem:

Theorem 4.1.2 *(First-Order Necessary Conditions) If α^* is a local minimizer and φ is continuously differentiable in an open neighborhood of α^*, then $\nabla\varphi(\alpha^*) = 0$.*

Proof Let us prove this by contradiction. Suppose that $\nabla\varphi(\alpha^*) \neq 0$. Let us define the vector $s = -\nabla\varphi(\alpha^*)$ and note that $s^T\nabla\varphi(\alpha^*) = -||\nabla\varphi(\alpha^*)|| < 0$. Because $\nabla\varphi$ is continuous near α^*, there is a scalar $T > 0$ such that

$$s^T\nabla\varphi(\alpha^* + \xi s) < 0, \qquad \text{for all } \xi \in [0, T].$$

For any $\bar{\xi} \in (0, T]$, we have by Taylor's theorem that

$$\varphi(\alpha^* + \bar{\xi}s) = \varphi(\alpha^*) + \bar{\xi}s^T\nabla\varphi(\alpha^* + \xi s), \qquad \text{for some } \xi \in (0, \bar{\xi}).$$

Therefore, $\varphi(\alpha^* + \bar{\xi}s) < \varphi(\alpha^*)$ for all $\bar{\xi} \in (0, T]$. We have found a direction leading away from α^* along which φ decreases, so α^* is not a local minimizer. This is a contradiction. □

We call α^* a **stationary point** if $\nabla\varphi(\alpha^*) = 0$. Hence, according to Theorem 4.1.2, any local minimizer must be a stationary point. In addition, when the objective function is convex, local and global minimizers are simple to characterize. We have the following:

Theorem 4.1.3 *When φ is convex, any local minimizer α^* is a global minimizer φ. If in addition φ is differentiable, then any stationary point α^* is a global minimizer of φ.*

4.2 Descent methods

The last several decades have seen the development of powerful optimization methods for optimization problems [16]. Typically, optimization methods are iterative algorithms for finding solutions of minimization problems. Here, the user supplies a starting point α_0 and the algorithm generates a sequence of iterates $\{\alpha_k\}$ that terminates when either a minimizer has been approximated by some sufficiency criteria or when no more progress can be made. Among such methods, a general class of methods known to be globally convergent are the **descent methods** [10]. The idea of descent methods is to find, at the current k^{th} iterate α_k, a direction s_k such that $\varphi(\alpha_k + \xi s_k)$ is decreasing at $\xi = 0$. To decide how to move from one iterate to the next, the descent method uses information about the function φ at α_k and possibly also information about $\alpha_0, \ldots, \alpha_{k-1}$ and find a new iterate α_{k+1} giving a smaller function value than α_k. An algorithm of a general descent method is presented in Algorithm 2.

Algorithm 2 General descent method

Choose an initial point α_0.
for $k = 0, 1, \ldots$ **do**
  if $\nabla\varphi(\alpha_k) = 0$ **then**
    Stop.
  end if
  Choose a descent direction s_k.
  Choose a step size $\delta_k > 0$.
  Set $\alpha_{k+1} = \alpha_k + \delta_k s_k$.
end for

4.2.1 Choice of Search Direction

In using the descent method, one major consideration needed to be taken is the choice of the descent direction. The steepest descent direction $-\nabla\varphi(\alpha_k)$ is the most obvious choice for search direction since intuitively, among all the directions we could move from α_k, it is the one along φ decreases most rapidly. Further, if $s_k = -\nabla\varphi(\alpha_k) =: -\nabla\varphi_k$, then

$$s_k^T \nabla\varphi_k = -\,||\nabla\varphi(\alpha_k)|| < 0,$$

and this property guarantees that the function φ can be reduced along this direction along s_k. Hence, we have an algorithm that finds a local minimum of a function using gradient

descent where one takes steps proportional to the negative of the gradient of the function at the current point. This is called the **method of steepest descent** or **gradient descent.**

4.2.2 Choice of Step Length

The success of the descent method does not only depend on appropriate and effective choice of the descent direction s_k. It also depends on the choice of step length δ_k for each iteration. In the descent method, it is an important consideration to decide how far to move along the descent direction. In choosing the step length δ_k, we aim that the resulting δ_k gives a substantial decrease of φ but at the same time we do not want to spend too much effort on coming up with the choice. The best choice for δ_k is obviously is

$$\delta_k = \underset{\delta>0}{\arg\min} \;\; \varphi(\alpha_k + \delta s_k),$$

but this is expensive to compute. However, a practical strategy is to perform an inexact approximate to this giving φ sufficient reduction where the computation is less expensive.

An inexact approximation providing an efficient way of computing an acceptable step length δ_k that reduces the cost in some measure of sufficiency is stipulated by the Wolfe conditions. A step length δ_k is said to satisfy the Wolfe conditions if the following two inequalities hold:

$$\varphi(\alpha_k + \delta_k s_k) \leq \varphi(\alpha_k) + c_1 \delta_k s_k^T \nabla\varphi(\alpha_k), \tag{4.1}$$

$$s_k^T \nabla\varphi(\alpha_k + \delta_k s_k) \geq c_2 s_k^T \nabla\varphi(\alpha_k), \tag{4.2}$$

with $0 < c_1 < c_2 < 1$. Inequality (4.1) is known as the **sufficient decrease** or the **Armijo condition** which ensures that δ_k decreases φ sufficiently and (4.2) is the **curvature condition** which makes sure that the slope of the function $\varphi(\alpha_k + \delta s_k)$ at δ_k is greater than c_2 times that at $\delta = 0$.

However, if the algorithm chooses its candidates appropriately, we can dispense (4.2) and just use the sufficient decrease condition to terminate the search for the proper step length. This is done through the **backtracking approach** (see [16]). For clarity, from what we gathered, we present in Algorithm 3 the algorithm of the backtracking steepest descent method using the Armijo condition that we shall adapt in our implementation.

Algorithm 3 Steepest Descent for minimizing φ

Set iteration counter $k = 0$ and initial guess for the control $\alpha_0 = (\alpha_0^{(x)}, \alpha_0^{(y)})$.

while $||\nabla\varphi(\alpha_k)|| > \text{tol}$, **do**

Compute the descent direction s_k, we take $s_k = -\nabla\varphi(\alpha_k)$.

Select a step length δ_k to somehow minimize $\varphi(\alpha_k + \delta s_k)$ over $\delta \in \mathbb{R}$. We do this by employing the Backtracking Line Search Method which uses the Sufficient Decrease/ Armijo's condition. Set iteration counter $j = 0$, initial step length guess $\delta^0 > 0$ and choose $\omega \in (0, 1)$ and $\kappa \in (0, 1)$.

while $\varphi(\alpha_k + \delta^j s_k) > \varphi(\alpha_k) + \kappa\delta^j s_k^T \nabla\varphi(\alpha_k)$ $(= \varphi(\alpha_k) + \kappa\delta^j(-||\nabla\varphi(\alpha_k)||^2))$, **do**

$\delta^{j+1} = \omega\delta^j$.
$j = j + 1$.

end while

Set $\delta_k = \delta^j$.

Update $\alpha_{k+1} = \alpha_k + \delta_k s_k$.

$k = k + 1$.

end while

4.3 Implementation, Results and Discussions

In this section we implement our algorithm and present and analyze the obtained results. We adapt notations as they were in Subsection 3.4.1. For our implementation, we apply the following numerical integration rule for the evaluation of the objective function and the gradient norm. For the objective function φ, we have

$$\begin{aligned}
\varphi(w_t,\alpha_t) &= \frac{1}{2}\int_{\Omega\times\mathcal{T}}\left|w_t - w_t^d\right|^2 + \frac{\lambda}{2}\int_{\Omega\times\mathcal{T}}|\alpha_t|^2 \\
&= \frac{1}{2}\int_{\Omega\times\mathcal{T}}\left(u_t^{(x)} - u_t^{(x)d}\right)^2 + \left(u_t^{(y)} - u_t^{(y)d}\right)^2 + \left(p_t - p_t^d\right)^2 \\
&\quad + \frac{\lambda}{2}\int_{\Omega\times\mathcal{T}}\alpha_t^{(x)2} + \alpha_t^{(y)2} \\
&= \frac{1}{2}\frac{1}{T-1}\frac{1}{m-1}\frac{1}{n-1}\sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n}\Bigg(\Bigg[\left(u_t^{(x)} - u_t^{(x)d}\right)^2 \\
&\quad + \left(u_t^{(y)} - u_t^{(y)d}\right)^2 + \left(p_t - p_t^d\right)^2\Bigg] + \lambda\left[\alpha_t^{(x)2} + \alpha_t^{(y)2}\right]\Bigg)
\end{aligned}$$

where $w_t \equiv w((x_i,y_j),t)$ and $\alpha_t \equiv \alpha((x_i,y_j),t)$. Further, the norm of the gradient is given by

$$\begin{aligned}
||g_t||_{L^2(\Omega\times\mathcal{T})} &= \left(\int_{\Omega\times\mathcal{T}}|g_t|^2\right)^{\frac{1}{2}} \\
&= \left(\int_{\Omega\times\mathcal{T}}\left[g_t^{(x)2} + g_t^{(y)2}\right]\right)^{\frac{1}{2}} \\
&= \left(\frac{1}{T-1}\frac{1}{m-1}\frac{1}{n-1}\sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[g_t^{(x)2} + g_t^{(y)2}\right]\right)^{\frac{1}{2}}
\end{aligned}$$

where $g_t \equiv g((x_i,y_j),t)$.

We first identify a desired state whose behavior is what we aim our simulated flow to achieve. As required by the algorithm to supplied by the user, we start with an initial guess for the forcing. We use the LBM as the (forward) solver of the states as we presented in Chapter 2. We then proceed with computing the adjoint states through the adjoint method (backward) solver allowing us to compute the gradient of the objective function, as presented in Chapter 3, giving us the descent direction enabling us to update the forcing iterate leading us to a new forcing, new control yielding a smaller value of the objective. This results to a new forcing which then requires another forward simulation of the states

and backward computation of the adjoint states leading to a new gradient. The process continues until we detect a stationary point.

4.3.1 Flow on an infinite channel

First, we test our implementation of the algorithm for control of fluid flows on an infinite channel. We simulate flows on a 50×40 lattice mesh. In this simulation, the periodic boundary conditions are used on the inlet and outlet of our constructed lattice and the bounce-back boundary conditions are used for the no-slip boundary conditions on the channel walls. Assuming a fluid on an infinite channel with kinematic viscosity of 1 throughout time $t = 1, \ldots, 100$ where the fluid has a zero initial velocity, we consider the states of a flow driven by a constant volume force of $\alpha = (0.1, 0)$. Let us refer to the generated states as the desired states for the optimal control problem.

An interesting inquiry for the optimal control problem to test the algorithm would be: on a fluid of a kinematic viscosity of 1, what driving force, constant throughout all the nodes of the domain and throughout each time $t = 1, \ldots, 100$, should one employ so as to influence the behavior of the states of the simulation for it to be as "close" as possible to those of the desired states. In other words, this problem tests whether the algorithm finds the solution that we already know which is obviously $\alpha = (0.1, 0)$. We use a stopping criteria demand the program to terminate when $||\nabla\varphi|| \leq 10^{-7}$, machine precision, is assigned.

Favorable results are obtained and are presented in Table 4.1. An arbitrary starting point of $\alpha_0 = (-0.09, -0.0015)$ for the iteration procedure of steepest descent was chosen. It is apparent from this table that there is a clear trend of decreasing for the values of φ and $||\nabla\varphi||$ towards zero. This is expected since we want to minimize φ and due to the first-order necessary condition, if we are approaching the minimum, then $||\nabla\varphi|| \to 0$. The number of Armijo sub-iterations are also given on the table. For each main iteration of the steepest descent method, we set an initial guess $\delta = 10$ for the step length and the Armijo condition determines which step length gives a sufficient decrease for the objective function by reducing it a reduction rate of ω assigned to be 0.1 until the appropriate step length is determined and identified as δ_k. Observe that for the selected k^{th} iterates shown in the table, 2 Armijo sub-iterations were performed. This means $\delta_k = 0.1$ for these values of k provided in the table. The program terminates when a stationary point is sufficiently approximated at $k = 58$ as imposed by the stopping criteria. Affirmative results shown clearly tells that the iterates $\{\alpha_k\}$ for the control converges to $\alpha = (0.1, 0)$. Figure 4.1 visualizes the generated $u_t^{(x)}$ for certain obtained iterates of the steepest descent method with respect to the desired $u_t^{(x)}$. We visualize and observe the behavioral trend of the velocities $u_t^{(x)}$ as the effect of the control for $k = 1, 5, 10$ and 58.

φ	$\|\|\nabla\varphi\|\|$	$\alpha^{(x)}$	$\alpha^{(y)}$	k	armijo steps
142.3	0.48066	-0.09	-0.0015	1	
89.896	0.37898	-0.051008	-0.0016388	2	2
56.826	0.30032	-0.020013	-0.0016775	3	2
35.944	0.2387	0.0046295	-0.0016277	4	2
22.745	0.19002	0.024223	-0.0015118	5	2
2.3042	0.060907	0.076038	-0.00068683	10	2
0.23135	0.019375	0.092421	-0.00022236	15	2
0.023129	0.0061354	0.097601	-6.4442e-005	20	2
0.0023114	0.0019405	0.09924	-1.803e-005	25	2
0.00023113	0.00061374	0.099759	-4.9887e-006	30	2
2.3129e-005	0.00019416	0.099924	-1.3754e-006	35	2
2.3162e-006	6.1438e-005	0.099976	-3.788e-007	40	2
2.3245e-007	1.9446e-005	0.099992	-1.0428e-007	45	2
2.3725e-008	6.1557e-006	0.099998	-2.8707e-008	50	2
2.8085e-009	1.9489e-006	0.099999	-7.9019e-009	55	2
1.9493e-009	1.5485e-006	0.099999	-6.1049e-009	56	2
1.4069e-009	1.2303e-006	0.1	-4.7166e-009	57	2
1.0645e-009	9.7753e-007	0.1	-3.644e-009	58	2

Table 4.1: Sequence of iterates generated by the steepest descent method for optimal control on an infinite channel assuming a starting control point $\alpha_0 = (0.09, -0.0015)$ where the solution is $\alpha = (0.1, 0)$

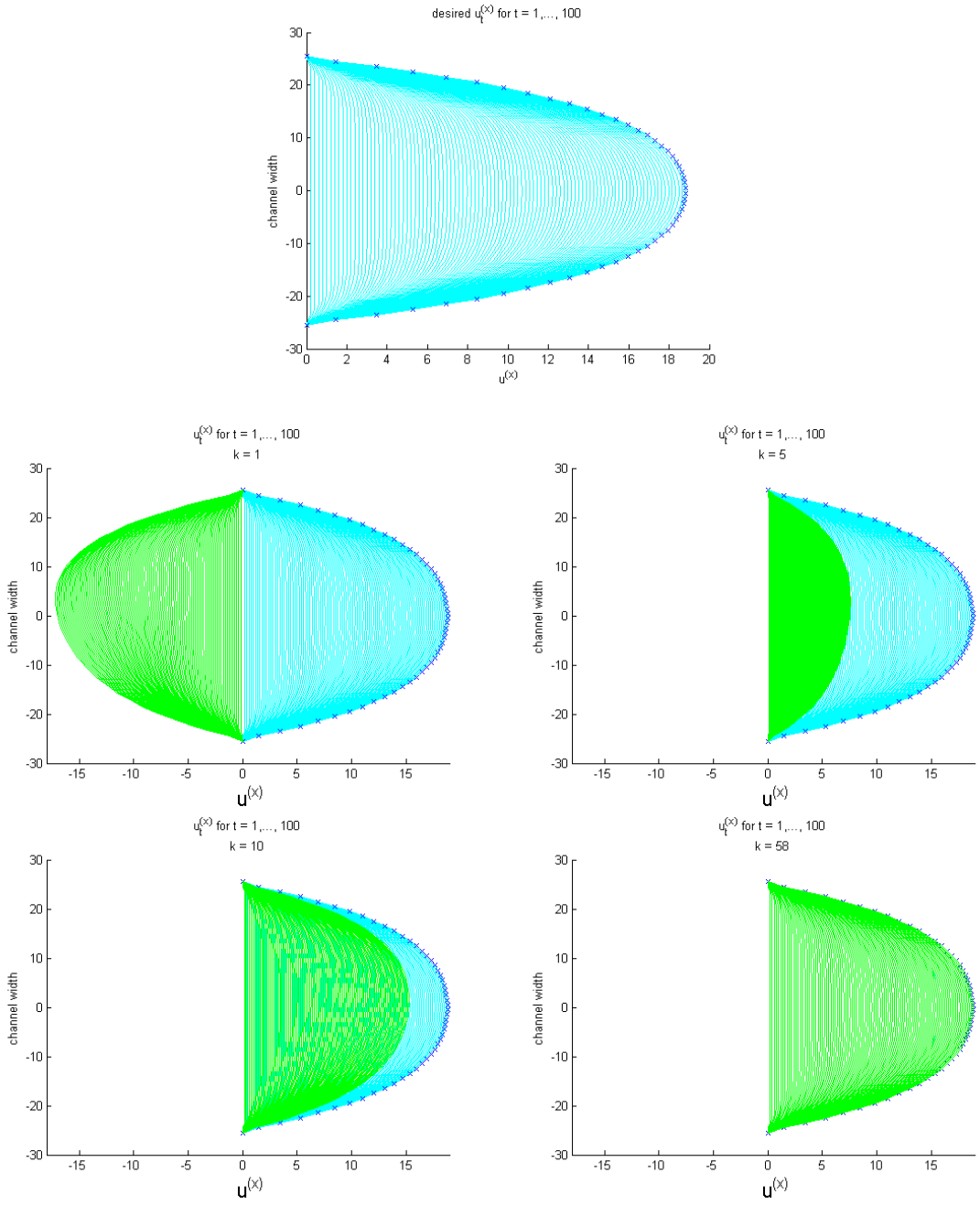

Figure 4.1: An illustration of the velocities $u_t^{(x)}$ as the effect of the obtained control for selected iterates of the steepest descent method $k = 1, 5, 10$ and 58

Figure 4.2 illustrates the contour map of the objective function over the subdomain $[0.04, 0.14] \times [0.015, -0.015]$. It is clear in the map that the objective attains its minimum at (0.1,0). The orange line traces the path of the control iterates obtained from steepest descent method with the points marked by $*$ are the control generated by the algorithm starting from the initial guess $\alpha = (-0.09, -0.0015)$ as it approach (0.1,0). Therefore, we see that method of steepest descent leads us to (0.1,0), i.e., to the point where the value of the function φ is minimal. It is important to note however that producing this map is extremely expensive since this requires evaluation of φ for a good number of control points to get an accurate shape of φ. As we have explained in the beginning of the chapter, we do not want to do such. This is being remedied by the steepest descent as it computes only necessary evaluations to reach the optimal control.

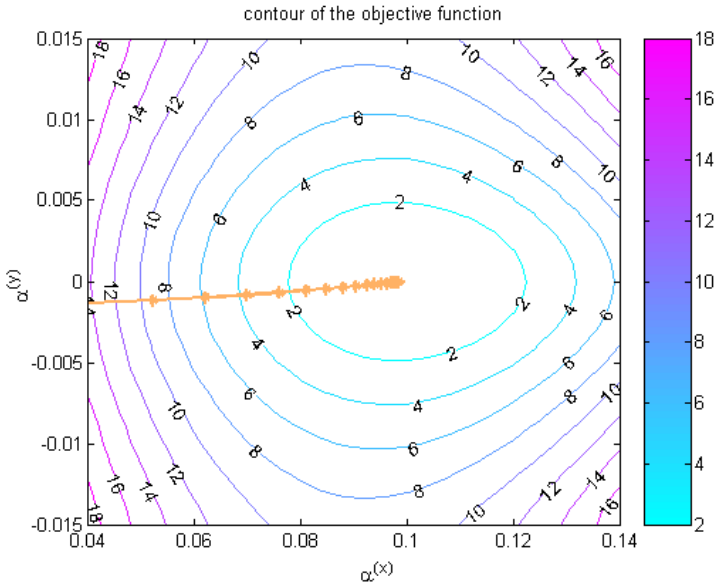

Figure 4.2: The contour map of the objective function for optimal control on an infinite channel where the solution is $\alpha = (0.1, 0)$

For the same desired case, we also test different initial guesses such as $\alpha_0 = (0.15, 0.0005)$, $\alpha_0 = (0, 0)$, etc. but are omitted here since they yield similar favorable results. For $\alpha_0 = (0.15, 0.0005)$, stationary point is approximated after 53 iterations, and for $\alpha_0 = (0, 0)$, after 56.

Now, we test the algorithm when the desired states are those generated when a zero forcing is applied. This suggests that we verify whether the algorithm finds the solution that is known to be $\alpha = (0, 0)$. An arbitrary starting point of $\alpha_0 = (0.2, -0.004)$ for the iteration procedure was chosen. Table 4.2 affirms that the iterates $\{\alpha_k\}$ for the control converges

φ	$\|\|\nabla\varphi\|\|$	$\alpha^{(x)}$	$\alpha^{(y)}$	k	armijo steps
156.83	0.53916	0.2	- 0.004	1	
99.794	0.41817	0.15949	-0.0038791	2	2
63.491	0.32738	0.12715	-0.0037145	3	2
40.38	0.25794	0.10135	-0.0035279	4	2
25.672	0.20407	0.080757	-0.0033335	5	2
2.66	0.064683	0.025839	-0.002436	10	2
0.28337	0.020653	0.0082287	-0.0017581	15	2
0.048343	0.0086509	-0.0034518	-0.00049388	20	2
0.0053951	0.0027595	-0.0010936	-0.00035558	25	2
0.00088621	0.0011539	0.00045943	-9.9497e-005	30	2
0.0001513	0.00048386	-0.00019306	-2.7835e-005	35	2
1.6903e-005	0.00015432	-6.1148e-005	-2.0039e-005	40	2
2.7745e-006	6.4539e-005	2.5695e-005	-5.6061e-006	45	2
4.7347e-007	2.7063e-005	-1.0797e-005	-1.5683e-006	50	2
5.2973e-008	8.6322e-006	-3.4199e-006	-1.1291e-006	55	2
8.6864e-009	3.6098e-006	1.4371e-006	-3.1586e-007	60	2
1.4816e-009	1.5136e-006	-6.0388e-007	-8.8365e-008	65	2
9.4564e-010	1.2035e-006	-4.7984e-007	-8.2744e-008	66	2
6.0599e-010	9.5716e-007	-3.8127e-007	-7.748e-008	67	2

Table 4.2: Sequence of iterates generated by the steepest descent method for optimal control on an infinite channel assuming a starting control point $\alpha_0 = (0.2, -0.004)$ where the solution is $\alpha = (0, 0)$

to $\alpha = (0, 0)$ as the values of φ and $||\nabla\varphi||$ shows a clear trend of decreasing towards zero.

4.3.2 Flow in a Cavity

We test our implementation of the algorithm for fluid flows in a cavity. We simulate flows on a 50×40 lattice mesh. Here, the bounce-back boundary conditions are used to stipulate the no-slip boundary conditions on all the 4 walls of the cavity. We assume that the fluid has a kinematic viscosity of 2.5. The desired states are those states from $T = 1, \ldots, 100$ generated when the fluid has a zero initial velocity and there is zero forcing applied.

We assign an initial guess $\alpha_0 = (0.0027, -0.0027)$ as the constant volume forcing throughout time as a starting point for the control. From this, the algorithm shall search for the control that will minimize the objective function. Clearly, the solution is $\alpha = (0, 0)$. Hence, we expect the algorithm to generate a sequence of iterates $\{\alpha_k\}$ for the control that converges to $\alpha = (0, 0)$. A stopping criteria commands the program to terminate when $||\nabla\varphi|| \leq 10^{-16}$, machine double precision, is assigned.

φ	$\|\|\nabla\varphi\|\|$	$\alpha^{(x)}$	$\alpha^{(y)}$	k	armijo steps
0.047676	0.0049905	0.0027	-0.0027	1	
0.0053097	0.0010622	-0.00079628	-0.0014286	2	1
0.0022448	0.00069076	-0.00057793	-0.00089992	3	1
0.00096238	0.00045234	-0.00041946	-0.00056657	4	1
0.00041934	0.00029863	-0.00030444	-0.00035648	5	1
8.7731e-006	4.3216e-005	-6.1322e-005	-3.4718e-005	10	1
2.7615e-007	7.6687e-006	-1.2354e-005	-3.2575e-006	15	1
1.0474e-008	1.4936e-006	-2.489e-006	-2.7987e-007	20	1
4.1931e-010	2.9882e-007	-5.015e-007	-1.8471e-008	25	1
1.6995e-011	6.0159e-008	-1.0105e-007	9.7796e-011	30	1
6.9036e-013	1.2125e-008	-2.0361e-008	4.0442e-010	35	1
2.8046e-014	2.4438e-009	-4.1027e-009	1.2024e-010	40	1
1.1391e-015	4.9251e-010	-8.267e-010	2.8131e-011	45	1
4.6261e-017	9.925e-011	-1.6658e-010	6.0614e-012	50	1
1.8785e-018	2e-011	-3.3565e-011	1.261e-012	55	1
7.6275e-020	4.0301e-012	-6.7634e-012	2.5807e-013	60	1
3.097e-021	8.1207e-013	-1.3628e-012	5.2403e-014	65	1
1.2575e-022	1.6363e-013	-2.7461e-013	1.0597e-014	70	1
5.1051e-024	3.2971e-014	-5.5331e-014	2.1404e-015	75	1
2.0746e-025	6.6468e-015	-1.115e-014	4.3337e-016	80	1
8.4545e-027	1.3421e-015	-2.2501e-015	8.9191e-017	85	1
4.4348e-027	9.72e-016	-1.6329e-015	6.5446e-017	86	1

Table 4.3: Sequence of iterates generated by the steepest descent method for optimal control on fluid in a cavity assuming a starting control point $\alpha_0 = (0.0027, -0.0027)$ where the solution is $\alpha = (0, 0)$

Table 4.3 shows the sequence of iterates for $\varphi, ||\nabla\varphi||$ and α generated by the steepest descent method for optimal control on fluid in a cavity where k is the iterate number. The number of Armijo sub-iterations are also given on the table. For each main iteration of the steepest descent method, we assign an initial guess $\delta = 10$ for the step length and the Armijo condition determines which step length gives a sufficient decrease for the objective function by reducing it a reduction rate of ω assigned to be 0.1 until the appropriate step length is determined and identified as δ_k. Table 4.3 provides us with the values of φ and it is apparent they decrease to zero as expected due to the Armijo condition guaranteeing the decrease in the objective value for each iteration. Observe that for the selected k^{th} iterate, 1 Armijo sub-iteration were performed. This means $\delta_k = 1$ for these values of k provided in the table. We also expect due to the first order necessary condition that $||\nabla\varphi||$ should approach zero. The program terminates when a stationary point is sufficiently approximated at $k = 86$.

Figure 4.3 illustrates the contour map of the objective function over the subdomain $[-0.003, 0.003]^2$. Evidently, we see that the objective attains its minimum at (0,0). The orange line traces path of the control iterates obtained from steepest descent method with the points marked by $*$ are the control generated by the algorithm starting from the initial guess $\alpha = (0.0027, -0.0027)$ as it approach (0,0). Therefore, we see that method of steepest descent leads us to (0,0), i.e., to the point where the value of the function φ is minimal. We note once again that this is expensive map is produced only for the reason that it may be interesting to demonstrate the trend of the iterates produced by the steepest descent method.

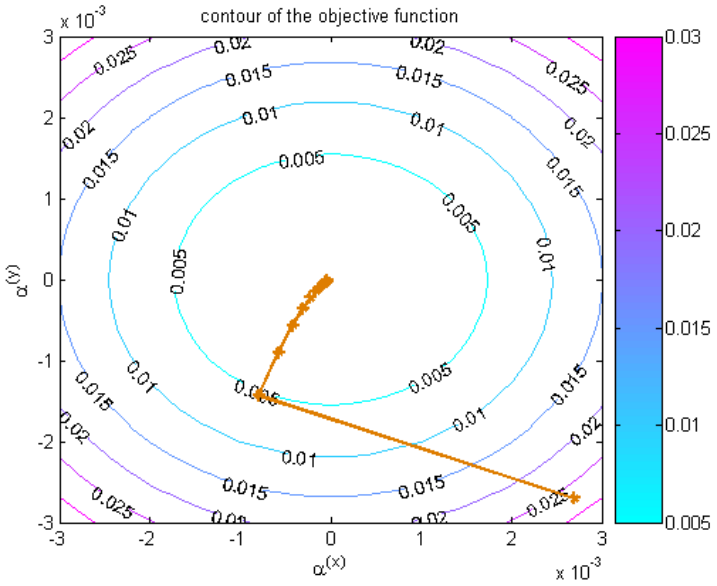

Figure 4.3: The contour map of the objective function for optimal control on fluid in a cavity where the solution is $\alpha = (0, 0)$

Chapter 5

Conclusion

A number of accomplishments have been achieved in this work. We have studied the Lattice Boltzmann method and examined its treatment that sets it apart from the typically-used schemes of simulating fluid flow and solving the incompressible Navier-Stokes equation. We have seen that the LBM uses molecular particle populations called distribution functions to model the space and time-averaged microscopic movements of fluid particles, and is governed by streaming and collision process where the collision rule is set so that the incompressible Navier-Stokes equations are satisfied. We have considered 2-dimensional fluid flows driven by an external force and introduced a simple forcing term to the LBM evolution equation. We have used the D2Q9 LBM model, a 2-dimensional LBM model that makes use of 9-velocity directions, and fluid have been simulated for cases on flow on an infinite channel and flow on a cavity, differing in the boundary conditions applied.

We have defined an optimal control problem on fluids composed of the state (the macroscopic properties of fluid), the control (the external forcing), the objective function, which indicates a measure of the deviation of the behavior of a simulated flow with respect to a desired flow behavior, and the constraint (the incompressible Navier-Stokes equation). We have also introduced the treatment we want to employ in solving the optimal control problem which is to "discretize then optimize." This suggests that we minimize an objective function such that the LBM solver serves as the discretized representation of the optimization constraint. The work has also given an account of and the reasons for the use of the discrete adjoint method. We have seen that the discrete adjoint method allows efficient evaluation of the derivative (the gradient) of the objective function that depends on the solution of a discretized PDE where the derivative is computed inexpensively by introducing a set of variables called the adjoint states that is computed through a backward solver derived from the LBM forward-solver.

The negative of the computed gradient has then been used as the choice of the direction of descent in the descent method for finding the optimal control - the local minimizer of the objective function. An algorithm has been proposed and implemented. As a test case, we have performed the optimization to find the optimal control, constant throughout all the nodes of the lattice and throughout time that yields the simulated flow that best behaves like the desired states. We have investigated and assessed the application of the method of steepest descent for flows on an infinite channel and flows in a cavity.

Above all, this explorative study has developed insights and understanding to the "discretize then optimize" attack on an optimal control problem employing the LBM to compute states forward time and an LBM-type solver to compute adjoint states backward in time. Competencies, in comparison to other techniques, and limitations of this numerical scheme shall lead to a better understanding of its implications on various applications and lead the way to further research for the improvement and extension of such scheme.

5.1 Limitations and recommendations for further work

However, a number of limitations restrains the results of this work and opens up to extensions for further work and investigation. First, we have only surveyed flows in 2-dimensions with the use of D2Q9 model. Note that other models are also available in 2D and 3D. Many authors proposed modifications for the bounce-back boundary condition, (e.g. in [24] and [11]) for better accuracy, hence they may also be considered since we have only used a simple reflection rule in our implementaion. We have also made an assumption of a convenient and simplified form of the forcing term which we use as the control in the optimization problem. Different representation of the forcing term have also been proposed (e.g. in [7] or [23]), thus, investigations may be needed to examine the effect of such forcing term in accuracy and stability of the LBM with regards to satisfying the incompressible Navier-Stokes equation. Further work needs to be done to establish the most suitable forcing term depending on the dictates of the application. Moreover, we have to note that modifying the forcing term will change the adjoint equation. Also, for the optimal control problem, the study was limited by a control that is constant throughout space and time. Other objective functions can also be considered. Furthermore, the steepest descent method has also been criticized for taking too long to converge to a local minimum. Note that several works have proposed techniques to accelerate the steepest descent (e.g. [1] and [17]) and other methods apart from the steepest descent method can also be performed (see [16]). Finally, it will also be interesting to compare our findings to the results of the "optimize then discretize" track, and possibly, to the results of the other Navier-Stokes equations solvers apart from the LBM.

Bibliography

[1] N. Andrei, *An acceleration of gradient descent algorithm with backtracking for unconstrained optimization*. Numer. Algor. 42 (2006). 63-73.

[2] A. Bronstein, M. Bronstein, *Numerical Optimization*. Course material in Numerical geometry of nonrigid shapes. Department of Computer Science, Technion - Israel Institute of Technology. <http://www.cs.technion.ac.il/ mbron/index.html>

[3] A. Caiazzo, *Asymptotic Analysis of lattice Boltzmann method for Fluid-Structure interaction problems*. Ph.D. diss., Scuola Normale Superiore Pisa, TU Kaiserslautern, 2006.

[4] S. Chen, G.D. Doolen, *Lattice Boltzmann method for fluid flows*. Annual Rev. of Fluid Mech. 30 (1998). 329-364

[5] S.T. Engler, *Benchmarking the 2D Lattice Boltzmann BGK Model*. Course material in Complex Systems Simulations. Amsterdam Center for Computational Science, Universiteit van Amsterdam. <http://staff.science.uva.nl/ sloot/CSS/>

[6] M. Giles, N. Pierce, *An Introduction to the Adjoint Approach to Design*. Applied Scientific Research. 65 (2000). 393-415.

[7] Z. Guo, C. Zheng, B. Shi, *Discrete lattice effects on the forcing term in the lattice Boltzmann method*. Phys. Rev. E 65 (2002). 026701:1-12.

[8] X. He, L.-S. Luo, *Lattice Boltzmann Model for the Incompressible NavierStokes Equation*. J. Stat. Phys. 88 (1997). 927-944

[9] X. He, L.-S. Luo, *Theory of the lattice Boltzmann method: From the Boltzmann equation to the lattice Boltzmann equation*, Phys. Rev. E 56 (1997). 6811-6817.

[10] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, *Lecture Notes of the Autumn School Modelling and Optimization with Partial Differential Equations*. Hamburg, 26-30 September 2005.

[11] T. Inamuro, M. Yoshino, F. Ogino, *A non-slip boundary condition for lattice Boltzmann simulations*. Phys. Fluids. 7 (1995). 2928-2930.

[12] M. Junk, A. Klar, L.-S. Luo, *Asymptotic analysis of the lattice Boltzmann Equation*. Journal Comp. Phys. 210 (2005). 676-704.

[13] *LBM: Lattice Boltzmann Method*. 18 May 2008. <http://www.lbmethod.org/>

[14] L.-S. Luo, *Lattice Boltzmann Methods for Computational Fluid Dynamics*. National Institute of Aerospace, Virginia, USA. <http://research.nianet.org/ luo>

[15] A. McNamara, A. Treuille, Z. Popovic, J. Stam, *Fluid control using the adjoint method*. ACM Transactions on Graphics. 23 (2004). 449-456.

[16] J. Nocedal, S.J. Wright, *Numerical Optimization*. Springer, New York. 2nd ed. 2006.

[17] F.A. Potra, Y. Shi, *Efficient line search algorithm for unconstrained optimization*. J. Opt. Theory and Applications. 85 (1995). 677-704.

[18] Y.H. Qian, D. d'Humieres, P. Lallemand, *Lattice BGK Models for the Navier-Stokes equation*. Europhys. Letters. 17 (1992). 479-484.

[19] R. Schneider, *Applications of the Discrete Adjoint Method in Computational Fluid Dynamics*. Ph.D. diss., The University of Leeds, 2006.

[20] R.K. Standish, *On Various Questions in Nonequilibrium Statistical Mechanics Relating to Swarms and Fluid Flow*. 30 Oct 1997. <http://www.hpcoders.com.au/docs/thesis.pdf>

[21] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*. Oxford University Press. 2001.

[22] F.J. Vesely, *Statistical Physics: Course material with JAVA applets*. Institute of Experimental Physics, University of Vienna. 25 Jan 2005. <http://homepage.univie.ac.at/Franz.Vesely/sp_english/sp/sp.html>

[23] T. Wang, J. Wang, *Two-fluid model based on the lattice Boltzmann equation*. Phys. Rev. E 71 (2005). 045301:1-4.

[24] D.P. Ziegler, *Boundary conditions for lattice Boltzmann simulations*. J. Stat. Phys. 71 (1993). 1171-1177.

[25] Q. Zou, X. He, *On pressure and velocity boundary conditions for the lattice Boltzmann BGK model*. Phys. Fluids. 9 (1997). 1591-1598.

Appendix

A.1 Full details of the computations of the adjoint states and gradient

Following the proposed model of [8] for the incompressible Navier Stokes equation, we have here the LBM model of the flow on an infinite channel full detailed in the following anatomization:

The macroscopic properties are given by

$$\begin{aligned} u^{(x)}((x_i,y_j),t) &= p_1((x_i,y_j),t) - p_3((x_i,y_j),t) + p_5((x_i,y_j),t) - p_6((x_i,y_j),t) - p_7((x_i,y_j),t) + p_8((x_i,y_j),t) \\ u^{(y)}((x_i,y_j),t) &= p_2((x_i,y_j),t) - p_4((x_i,y_j),t) + p_5((x_i,y_j),t) + p_6((x_i,y_j),t) - p_7((x_i,y_j),t) - p_8((x_i,y_j),t) \\ p((x_i,y_j),t) &= \sum_{\hat{i}=1}^{9} p_{\hat{i}}((x_i,y_j),t). \end{aligned} \tag{A.1}$$

The streaming process after collision and forcing is depicted by

$$
\begin{aligned}
p_1((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_1((x_{i-1},y_j),t-1) \;+\; \tfrac{1}{\tau}p_1^{\mathrm{eq}}((x_{i-1},y_j),t-1) + \tfrac{1}{3}\alpha^{(x)}((x_{i-1},y_j),t-1) && i=2,\ldots,n, \quad j=1,\ldots,m,\\
p_2((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_2((x_i,y_{j-1}),t-1) \;+\; \tfrac{1}{\tau}p_2^{\mathrm{eq}}((x_i,y_{j-1}),t-1) + \tfrac{1}{3}\alpha^{(y)}((x_i,y_{j-1}),t-1) && i=1,\ldots,n, \quad j=2,\ldots,m,\\
p_3((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_3((x_{i+1},y_j),t-1) \;+\; \tfrac{1}{\tau}p_3^{\mathrm{eq}}((x_{i+1},y_j),t-1) - \tfrac{1}{3}\alpha^{(x)}((x_{i+1},y_j),t-1) && i=1,\ldots,n-1, \quad j=1,\ldots,m,\\
p_4((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_4((x_i,y_{j+1}),t-1) \;+\; \tfrac{1}{\tau}p_4^{\mathrm{eq}}((x_i,y_{j+1}),t-1) - \tfrac{1}{3}\alpha^{(y)}((x_i,y_{j+1}),t-1) && i=1,\ldots,n, \quad j=1,\ldots,m-1,\\
p_5((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_5((x_{i-1},y_{j-1}),t-1) \;+\; \tfrac{1}{\tau}p_5^{\mathrm{eq}}((x_{i-1},y_{j-1}),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_{i-1},y_{j-1}),t-1) + \alpha^{(y)}((x_{i-1},y_{j-1}),t-1)) && i=2,\ldots,n, \quad j=2,\ldots,m,\\
p_6((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_6((x_{i+1},y_{j-1}),t-1) \;+\; \tfrac{1}{\tau}p_6^{\mathrm{eq}}((x_{i+1},y_{j-1}),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_{i+1},y_{j-1}),t-1) + \alpha^{(y)}((x_{i+1},y_{j-1}),t-1)) && i=1,\ldots,n-1, \quad j=2,\ldots,m,\\
p_7((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_7((x_{i+1},y_{j+1}),t-1) \;+\; \tfrac{1}{\tau}p_7^{\mathrm{eq}}((x_{i+1},y_{j+1}),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_{i+1},y_{j+1}),t-1) - \alpha^{(y)}((x_{i+1},y_{j+1}),t-1)) && i=1,\ldots,n-1, \quad j=1,\ldots,m-1,\\
p_8((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_8((x_{i-1},y_{j+1}),t-1) \;+\; \tfrac{1}{\tau}p_8^{\mathrm{eq}}((x_{i-1},y_{j+1}),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_{i-1},y_{j+1}),t-1) - \alpha^{(y)}((x_{i-1},y_{j+1}),t-1)) && i=2,\ldots,n, \quad j=1,\ldots,m-1,\\
p_9((x_i,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_9((x_i,y_j),t-1) \;+\; \tfrac{1}{\tau}p_9^{\mathrm{eq}}((x_i,y_j),t-1) && i=1,\ldots,n, \quad j=1,\ldots,m.
\end{aligned}
\tag{A.2}
$$

The periodic and the bounce back boundary conditions are taken care of

$$
\begin{aligned}
p_1((x_1,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_1((x_n,y_j),t-1) \;+\; \tfrac{1}{\tau}p_1^{\mathrm{eq}}((x_n,y_j),t-1) + \tfrac{1}{3}\alpha^{(x)}((x_n,y_j),t-1) && j=1,\ldots,m,\\
p_4((x_i,y_m),t) &= \left(1-\tfrac{1}{\tau}\right) p_2((x_i,y_m),t-1) \;+\; \tfrac{1}{\tau}p_2^{\mathrm{eq}}((x_i,y_m),t-1) + \tfrac{1}{3}\alpha^{(y)}((x_i,y_m),t-1) && i=1,\ldots,n,\\
p_3((x_n,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_3((x_1,y_j),t-1) \;+\; \tfrac{1}{\tau}p_3^{\mathrm{eq}}((x_1,y_j),t-1) - \tfrac{1}{3}\alpha^{(x)}((x_1,y_j),t-1) && j=1,\ldots,m,\\
p_2((x_i,y_1),t) &= \left(1-\tfrac{1}{\tau}\right) p_4((x_i,y_1),t-1) \;+\; \tfrac{1}{\tau}p_4^{\mathrm{eq}}((x_i,y_1),t-1) - \tfrac{1}{3}\alpha^{(y)}((x_i,y_1),t-1) && i=1,\ldots,n,\\
p_7((x_i,y_m),t) &= \left(1-\tfrac{1}{\tau}\right) p_5((x_i,y_m),t-1) \;+\; \tfrac{1}{\tau}p_5^{\mathrm{eq}}((x_i,y_m),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_i,y_m),t-1) + \alpha^{(y)}((x_i,y_m),t-1)) && i=1,\ldots,n,\\
p_5((x_1,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_5((x_n,y_{j-1}),t-1) \;+\; \tfrac{1}{\tau}p_5^{\mathrm{eq}}((x_n,y_{j-1}),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_n,y_{j-1}),t-1) + \alpha^{(y)}((x_n,y_{j-1}),t-1)) && j=2,\ldots,m,\\
p_8((x_i,y_m),t) &= \left(1-\tfrac{1}{\tau}\right) p_6((x_i,y_m),t-1) \;+\; \tfrac{1}{\tau}p_6^{\mathrm{eq}}((x_i,y_m),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_i,y_m),t-1) + \alpha^{(y)}((x_i,y_m),t-1)) && i=1,\ldots,n,\\
p_6((x_n,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_6((x_1,y_{j-1}),t-1) \;+\; \tfrac{1}{\tau}p_6^{\mathrm{eq}}((x_1,y_{j-1}),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_1,y_{j-1}),t-1) + \alpha^{(y)}((x_1,y_{j-1}),t-1)) && j=2,\ldots,m,\\
p_5((x_i,y_1),t) &= \left(1-\tfrac{1}{\tau}\right) p_7((x_i,y_1),t-1) \;+\; \tfrac{1}{\tau}p_7^{\mathrm{eq}}((x_i,y_1),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_i,y_1),t-1) - \alpha^{(y)}((x_i,y_1),t-1)) && i=1,\ldots,n,\\
p_7((x_n,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_7((x_1,y_{j+1}),t-1) \;+\; \tfrac{1}{\tau}p_7^{\mathrm{eq}}((x_1,y_{j+1}),t-1) + \tfrac{1}{3}(-\alpha^{(x)}((x_1,y_{j+1}),t-1) - \alpha^{(y)}((x_1,y_{j+1}),t-1)) && j=1,\ldots,m-1,\\
p_6((x_i,y_1),t) &= \left(1-\tfrac{1}{\tau}\right) p_8((x_i,y_1),t-1) \;+\; \tfrac{1}{\tau}p_8^{\mathrm{eq}}((x_i,y_1),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_i,y_1),t-1) - \alpha^{(y)}((x_i,y_1),t-1)) && i=1,\ldots,n,\\
p_8((x_1,y_j),t) &= \left(1-\tfrac{1}{\tau}\right) p_8((x_n,y_{j+1}),t-1) \;+\; \tfrac{1}{\tau}p_8^{\mathrm{eq}}((x_n,y_{j+1}),t-1) + \tfrac{1}{3}(\alpha^{(x)}((x_n,y_{j+1}),t-1) - \alpha^{(y)}((x_n,y_{j+1}),t-1)) && j=1,\ldots,m-1.
\end{aligned}
\tag{A.3}
$$

The equilibrium distributions for the 9 velocity directions are given by

$$
\begin{aligned}
p_1^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[\rho((x_i,y_j),t-1)+3u^{(x)}((x_i,y_j),t-1)+\tfrac{9}{2}u^{(x)2}((x_i,y_j),t-1)-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_2^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[p((x_i,y_j),t-1)+3u^{(y)}((x_i,y_j),t-1)+\tfrac{9}{2}u^{(y)2}((x_i,y_j),t-1)-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_3^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[p((x_i,y_j),t-1)-3u^{(x)}((x_i,y_j),t-1)+\tfrac{9}{2}u^{(x)2}((x_i,y_j),t-1)-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_4^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[p((x_i,y_j),t-1)-3u^{(y)}((x_i,y_j),t-1)+\tfrac{9}{2}u^{(y)2}((x_i,y_j),t-1)-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_5^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[p((x_i,y_j),t-1)+3(u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))+\tfrac{9}{2}(u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))^2-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_6^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[p((x_i,y_j),t-1)+3(-u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))+\tfrac{9}{2}(-u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))^2-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_7^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[p((x_i,y_j),t-1)+3(-u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))+\tfrac{9}{2}(-u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))^2-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_8^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[p((x_i,y_j),t-1)+3(u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))+\tfrac{9}{2}(u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))^2-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right]\\
p_9^{\text{eq}}((x_i,y_j),t-1) &= \tfrac{4}{9}\left[p((x_i,y_j),t-1)-\tfrac{3}{2}(u^{(x)2}((x_i,y_j),t-1)+u^{(y)2}((x_i,y_j),t-1))\right].
\end{aligned}
\tag{A.4}
$$

We need, $G_t = \dfrac{\partial Q_t}{\partial w_{t-1}}(w_{t-1},\alpha_{t-1})$. Recall we have

$$w_t = Q_t(w_{t-1},\alpha_{t-1}).$$

We can write this as

$$
w_t = \begin{bmatrix} u_t^{(x)} \\ u_t^{(y)} \\ p_t \end{bmatrix} = \begin{bmatrix} Q_t^1(w_{t-1},\alpha_{t-1}) \\ Q_t^2(w_{t-1},\alpha_{t-1}) \\ Q_t^3(w_{t-1},\alpha_{t-1}) \end{bmatrix}, \tag{A.5}
$$

and thus,

$$G_t = \frac{\partial Q_t}{\partial w_{t-1}} = \begin{bmatrix} \frac{\partial Q_t^1}{\partial u_{t-1}^{(x)}} & \frac{\partial Q_t^1}{\partial u_{t-1}^{(y)}} & \frac{\partial Q_t^1}{\partial p_{t-1}} \\ \frac{\partial Q_t^2}{\partial u_{t-1}^{(x)}} & \frac{\partial Q_t^2}{\partial u_{t-1}^{(y)}} & \frac{\partial Q_t^2}{\partial p_{t-1}} \\ \frac{\partial Q_t^3}{\partial u_{t-1}^{(x)}} & \frac{\partial Q_t^3}{\partial u_{t-1}^{(y)}} & \frac{\partial Q_t^3}{\partial p_{t-1}} \end{bmatrix}. \tag{A.6}$$

Now let us compute the entries of G_t. For example, suppose we want to differentiate the system Q_t with respect to $u_{t-1}^{(x)}$, dealing with the chain of functions that can be traced along (A.1) to (A.4) we have for the macroscopic properties

$$\begin{aligned} \frac{\partial}{\partial u_{t-1}^{(x)}} u^{(x)}((x_i,y_j),t) &= \frac{\partial}{\partial u_{t-1}^{(x)}} p_1((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_3((x_i,y_j),t) + \frac{\partial}{\partial u_{t-1}^{(x)}} p_5((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_6((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_7((x_i,y_j),t) + \frac{\partial}{\partial u_{t-1}^{(x)}} p_8((x_i,y_j),t) \\ \frac{\partial}{\partial u_{t-1}^{(x)}} u^{(y)}((x_i,y_j),t) &= \frac{\partial}{\partial u_{t-1}^{(x)}} p_2((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_4((x_i,y_j),t) + \frac{\partial}{\partial u_{t-1}^{(x)}} p_5((x_i,y_j),t) + \frac{\partial}{\partial u_{t-1}^{(x)}} p_6((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_7((x_i,y_j),t) - \frac{\partial}{\partial u_{t-1}^{(x)}} p_8((x_i,y_j),t) \\ \frac{\partial}{\partial u_{t-1}^{(x)}} p((x_i,y_j),t) &= \sum_{i=1}^{9} \frac{\partial}{\partial u_{t-1}^{(x)}} p_i((x_i,y_j),t), \end{aligned} \tag{A.7}$$

for the distribution functions

$$
\begin{aligned}
\frac{\partial}{\partial u_{t-1}^{(x)}} p_1((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_1^{\mathrm{eq}}((x_{i-1},y_j),t-1) && i=2,\ldots,n, && j=1,\ldots,m,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_2((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_2^{\mathrm{eq}}((x_i,y_{j-1}),t-1) && i=1,\ldots,n, && j=2,\ldots,m,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_3((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_3^{\mathrm{eq}}((x_{i+1},y_j),t-1) && i=1,\ldots,n-1, && j=1,\ldots,m,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_4((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_4^{\mathrm{eq}}((x_i,y_{j+1}),t-1) && i=1,\ldots,n, && j=1,\ldots,m-1,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_5((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_5^{\mathrm{eq}}((x_{i-1},y_{j-1}),t-1) && i=2,\ldots,n, && j=2,\ldots,m,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_6((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_6^{\mathrm{eq}}((x_{i+1},y_{j-1}),t-1) && i=1,\ldots,n-1, && j=2,\ldots,m,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_7((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_7^{\mathrm{eq}}((x_{i+1},y_{j+1}),t-1) && i=1,\ldots,n-1, && j=1,\ldots,m-1,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_8((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_8^{\mathrm{eq}}((x_{i-1},y_{j+1}),t-1) && i=2,\ldots,n, && j=1,\ldots,m-1,\\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_9((x_i,y_j),t) &= \frac{1}{\tau}\frac{\partial}{\partial u_{t-1}^{(x)}} p_9^{\mathrm{eq}}((x_i,y_j),t-1) && i=1,\ldots,n, && j=1,\ldots,m,
\end{aligned}
\tag{A.8}
$$

for the periodic and the bounce back boundary conditions,

$$
\begin{aligned}
\frac{\partial}{\partial u_{t-1}^{(x)}} p_1((x_1, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_1^{\mathrm{eq}}((x_n, y_j), t-1) && j = 1, \ldots, m, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_4((x_i, y_m), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_2^{\mathrm{eq}}((x_i, y_m), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_3((x_n, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_3^{\mathrm{eq}}((x_1, y_j), t-1) && j = 1, \ldots, m, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_2((x_i, y_1), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_4^{\mathrm{eq}}((x_i, y_1), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_7((x_i, y_m), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_5^{\mathrm{eq}}((x_i, y_m), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_5((x_1, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_5^{\mathrm{eq}}((x_n, y_{j-1}), t-1) && j = 2, \ldots, m, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_8((x_i, y_m), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_6^{\mathrm{eq}}((x_i, y_m), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_6((x_n, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_6^{\mathrm{eq}}((x_1, y_{j-1}), t-1) && j = 2, \ldots, m, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_5((x_i, y_1), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_7^{\mathrm{eq}}((x_i, y_1), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_7((x_n, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_7^{\mathrm{eq}}((x_1, y_{j+1}), t-1) && j = 1, \ldots, m-1, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_6((x_i, y_1), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_8^{\mathrm{eq}}((x_i, y_1), t-1) && i = 1, \ldots, n, \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_8((x_1, y_j), t) &= \frac{1}{\tau} \frac{\partial}{\partial u_{t-1}^{(x)}} p_8^{\mathrm{eq}}((x_n, y_{j+1}), t-1) && j = 1, \ldots, m-1,
\end{aligned}
\tag{A.9}
$$

and for the equilibrium distributions for the 9 velocity directions, with respect to $u_{t-1}^{(x)}$,

$$
\begin{aligned}
\frac{\partial}{\partial u_{t-1}^{(x)}} p_1^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[3+9u^{(x)}((x_i,y_j),t-1)-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_2^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_3^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[-3+9u^{(x)}((x_i,y_j),t-1)-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_4^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{9}\left[-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_5^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[3+9(u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_6^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[-3-9(-u^{(x)}((x_i,y_j),t-1)+u^{(y)}((x_i,y_j),t-1))-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_7^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[-3-9(-u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_8^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{1}{36}\left[3+9(u^{(x)}((x_i,y_j),t-1)-u^{(y)}((x_i,y_j),t-1))-3u^{(x)}((x_i,y_j),t-1)\right] \\
\frac{\partial}{\partial u_{t-1}^{(x)}} p_9^{\mathrm{eq}}((x_i,y_j),t-1) &= \tfrac{4}{9}\left[-3u^{(x)}((x_i,y_j),t-1)\right].
\end{aligned}
\tag{A.10}
$$

To sum up what has been done, from $w_t = Q_t(w_{t-1}, \alpha_{t-1})$, differentiating (A.1) to (A.4) with respect to $u_{t-1}^{(x)}$, (A.7) gives us the expressions for the entries in the second row of G_t illustrated in (A.6) and the terms found in the said expressions can be traced along (A.7) to (A.10). The rest of the columns of G_t can be computed similarly by differentiating (A.1) to (A.4) with respect to $u_{t-1}^{(y)}$ and p_{t-1}, respectively. Ultimately, follows the adjoint equation given by

$$\underbrace{p_t}_{\in \mathbb{R}^{3\times 1}} = \underbrace{G_{t+1}}_{\in \mathbb{R}^{3\times 3}} \underbrace{p_{t+1}}_{\in \mathbb{R}^{3\times 1}} + \underbrace{\gamma_t}_{\in \mathbb{R}^{3\times 1}}, \qquad t = T-1, \ldots, 1. \tag{A.11}$$

Further, we take note that the adjoint states p_t are solved backward in time and $p_T = \gamma_T = w_T^{(x)} - w_T^{(x)^d}$.

We also need $H_t = \dfrac{\partial Q_t}{\partial \alpha_{t-1}}(w_{t-1}, \alpha_{t-1})$. Following the breakdown of Q_t given by (A.5), we have

$$H_t = \frac{\partial Q_t}{\partial \alpha_{t-1}} = \begin{bmatrix} \dfrac{\partial Q_t^1}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^1}{\partial \alpha_{t-1}^{(y)}} \\ \dfrac{\partial Q_t^2}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^2}{\partial \alpha_{t-1}^{(y)}} \\ \dfrac{\partial Q_t^3}{\partial \alpha_{t-1}^{(x)}} & \dfrac{\partial Q_t^3}{\partial \alpha_{t-1}^{(y)}} \end{bmatrix}. \tag{A.12}$$

Differentiating the system (A.1) to (A.4) with respect to $\alpha_{t-1} = (\alpha_{t-1}^{(x)}, \alpha_{t-1}^{(y)})$ (i.e. to each component of α_{t-1}), we obtain $H_t \in \mathbb{R}^{3\times 2}$. Let us take for example $\alpha_{t-1}^{(x)}$, the control force term along the x-direction. We differentiate $w_t = Q_t(w_{t-1}, \alpha_t)$ with respect to $\alpha_{t-1}^{(x)}$. We have for the macroscopic properties

$$\begin{aligned}
\frac{\partial}{\partial \alpha_{t-1}^{(x)}} u^{(x)}((x_i, y_j), t) &= \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_1((x_i, y_j), t) - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_3((x_i, y_j), t) + \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_5((x_i, y_j), t) \\
&\quad - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_6((x_i, y_j), t) - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_7((x_i, y_j), t) + \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_8((x_i, y_j), t) \\
\frac{\partial}{\partial \alpha_{t-1}^{(x)}} u^{(y)}((x_i, y_j), t) &= \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_2((x_i, y_j), t) - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_4((x_i, y_j), t) + \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_5((x_i, y_j), t) \\
&\quad + \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_6((x_i, y_j), t) - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_7((x_i, y_j), t) - \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_8((x_i, y_j), t) \\
\frac{\partial}{\partial \alpha_{t-1}^{(x)}} p((x_i, y_j), t) &= \sum_{i=1}^{9} \frac{\partial}{\partial \alpha_{t-1}^{(x)}} p_i((x_i, y_j), t),
\end{aligned} \tag{A.13}$$

for the distribution functions

$$
\begin{aligned}
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_1((x_i, y_j), t) &= \tfrac{1}{3} && i = 2, \ldots, n, && j = 1, \ldots, m, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_2((x_i, y_j), t) &= 0 && i = 1, \ldots, n, && j = 2, \ldots, m, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_3((x_i, y_j), t) &= -\tfrac{1}{3} && i = 1, \ldots, n-1, && j = 1, \ldots, m, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_4((x_i, y_j), t) &= 0 && i = 1, \ldots, n, && j = 1, \ldots, m-1, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_5((x_i, y_j), t) &= \tfrac{1}{3} && i = 2, \ldots, n, && j = 2, \ldots, m, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_6((x_i, y_j), t) &= -\tfrac{1}{3} && i = 1, \ldots, n-1, && j = 2, \ldots, m, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_7((x_i, y_j), t) &= -\tfrac{1}{3} && i = 1, \ldots, n-1, && j = 1, \ldots, m-1, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_8((x_i, y_j), t) &= \tfrac{1}{3} && i = 2, \ldots, n, && j = 1, \ldots, m-1, \\
\frac{\partial}{\partial \alpha_{t-1}^{(1)}} f_9((x_i, y_j), t) &= 0 && i = 1, \ldots, n, && j = 1, \ldots, m,
\end{aligned}
\tag{A.14}
$$

for the periodic and the bounce back boundary conditions we have

$$
\begin{aligned}
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_1((x_1, y_j), t) &= \tfrac{1}{3} && j = 1, \ldots, m, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_4((x_i, y_m), t) &= 0 && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_3((x_n, y_j), t) &= \tfrac{1}{3} && j = 1, \ldots, m, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_2((x_i, y_1), t) &= 0 && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_7((x_i, y_m), t) &= \tfrac{1}{3} && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_5((x_1, y_j), t) &= \tfrac{1}{3} && j = 2, \ldots, m, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_8((x_i, y_m), t) &= -\tfrac{1}{3} && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_6((x_n, y_j), t) &= -\tfrac{1}{3} && j = 2, \ldots, m, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_5((x_i, y_1), t) &= -\tfrac{1}{3} && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_7((x_n, y_j), t) &= -\tfrac{1}{3} && j = 1, \ldots, m-1, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_6((x_i, y_1), t) &= \tfrac{1}{3} && i = 1, \ldots, n, \\
\frac{\partial}{\partial \alpha^{(x)}_{t-1}} f_8((x_1, y_j), t) &= \tfrac{1}{3} && j = 1, \ldots, m-1,
\end{aligned}
\tag{A.15}
$$

for the equilibrium distributions for the 9 velocity directions,

$$\begin{aligned}\frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_1^{\mathrm{eq}}((x_i,y_j),t) &= \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_2^{\mathrm{eq}}((x_i,y_j),t) = \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_3^{\mathrm{eq}}((x_i,y_j),t) = \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_4^{\mathrm{eq}}((x_i,y_j),t) = \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_5^{\mathrm{eq}}((x_i,y_j),t) = \\ \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_6^{\mathrm{eq}}((x_i,y_j),t) &= \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_7^{\mathrm{eq}}((x_i,y_j),t) = \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_8^{\mathrm{eq}}((x_i,y_j),t) = \frac{\partial}{\partial\alpha_{t-1}^{(x)}}f_9^{\mathrm{eq}}((x_i,y_j),t) = 0.\end{aligned} \tag{A.16}$$

Thus, (A.13) gives us the expressions for the entries in the first row of H_t illustrated in (A.12) and the terms appearing on the said expressions can be traced along (A.13) to (A.16). The second column of H_t can be computed similarly by differentiating (A.1) to (A.4) with respect to $\alpha_{t-1}^{(y)}$.

Finally, the gradient $g_t \in \mathbb{R}^2$ of φ_t with respect to control $\alpha_t \in \mathbb{R}^2$ is given by the gradient equation

$$\underbrace{g_t}_{\in\mathbb{R}^{2\times1}} = \frac{d\varphi_t}{d\alpha_t} = \underbrace{H_t^T}_{\in\mathbb{R}^{2\times3}}\ \underbrace{p_t}_{\in\mathbb{R}^{3\times1}} + \underbrace{h_t}_{\in\mathbb{R}^{2\times1}}, \qquad t = T-1,\ldots,1. \tag{A.17}$$

Declaration

I hereby declare that I have done this thesis on my own and that no other sources than those listed have been used.

August 11, 2008, Kaiserslautern, Germany Vryan Gil S. Palma