A New Lattice Boltzmann Method

An honors thesis for the Department of Mathematics

Han Cheng Lie

Tufts University, 2010

Contents

Chapter 1

Introduction

This thesis is on a continuous-time Lattice Boltzmann Equation. It combines work in mathematical fluid mechanics and computer programming, and it was conceived after research work in the summer of 2009 on computing symbolic dynamics of turbulent fluid flow on the 2-torus. The common goal that unites the prior research work and this paper is a better understanding of turbulence using Lattice Boltzmann methods. In this paper we will focus on the Lattice Boltzmann aspect.

The field of mathematical fluid mechanics has enjoyed a long and productive history. Over 150 years, the mathematical theory of fluid dynamics has been refined and expanded upon, with the Navier-Stokes Equations

$$\begin{aligned} \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} &= -\nabla p + \nu \nabla^2 \boldsymbol{u} \\ \nabla \cdot \boldsymbol{u} &= 0 \end{aligned}$$

arguably being the centerpiece of fluid dynamics. Solutions to many problems remain elusive, however, with the phenomenon of turbulence being one of the earliest to be documented (as early as the Renaissance by Leonardo da Vinci) and the most difficult to understand. This paper will attempt to make a contribution to understanding turbulence by describing and analyzing a new Lattice Boltzmann method for computational fluid dynamics that may be used to find unstable periodic orbits of turbulent fluid flow.

Lattice Boltzmann methods are the product of two different ideas - the Boltzmann Equation from statistical physics and cellular automata from computer science. The Boltzmann Equation grew out of Ludwig Boltzmann's work on gas theory, and it was devised in order to capture the aggregate behavior of gas particles. Cellular automata are interesting computational tools that allow researchers to model the behavior of a system. The discovery by Frisch, Hasslacher and Pomeau that cellular automata could replicate some features of fluid flow eventually led to more advanced models such as the Lattice Boltzmann Methods, which used the Boltzmann Equation and cellular automata theory to model the behavior of a fluid accurately, that is, in agreement with the Navier-Stokes Equations. In this paper we will focus solely on the Incompressible Navier-Stokes Equations, which we may sometimes refer to as the 'Navier-Stokes Equations'.

1.1 Motivation

Computing symbolic dynamics of turbulent fluid flow on the 2-torus began with the idea that the state of a fluid at a point in time can be pictured as a point in some phase space. The phase space can be thought of as the set of all possible fluid states. Since we are interested in fluids that are governed by the Navier-Stokes Equations, one can imagine starting at some initial condition and integrating the Navier-Stokes equations forward in time. This corresponds to starting at a point in the phase space, and tracking that point as it moves along some path in the phase space.

The movement of the point through phase space reflects the changing state of the fluid as one integrates the Navier-Stokes Equations forward in time. Given this paradigm of tracking the trajectory of the fluid state in phase space, one might be interested in knowing when the trajectory crossed a point in the phase space - that is, when the fluid was at a particular state - and what properties the fluid had at that point. In order to compute symbolic dynamics of turbulent fluid flow, one wishes to know the state of the fluid to a high degree of accuracy.

Lattice Boltzmann Methods have hitherto been implemented in discrete time. This means that if one were to use a Lattice Boltzmann Method to model the behavior of a fluid accurately with respect to the Navier-Stokes Equations, one would effectively be constrained to knowing the behavior of the fluid in between regular time intervals of a fixed length. Discrete time Lattice Boltzmann Methods in and of themselves are very useful, but for computing symbolic dynamics, they are not sufficient.

Imagine that a researcher is trying to visually track the movement of a particle in a box. The box is illuminated by a strobe light that flashes on for an instant at fixed time intervals. When the strobe light flashes on, for that instant the researcher knows the particle's location and can obtain data about it, such as its kinetic energy. Then the strobe light turns off until it flashes on again a while later. During the interval in which the strobe light is off, the researcher knows nothing about the particle except that somehow the particle must move continuously between points in the box (i.e., there are no wormholes or singularities in space-time in the box). That still leaves an infinite number of ways that the particle could have moved from one location to another during the interval.

This is where the need for a continuous-time Lattice Boltzmann Equation becomes clear. With discrete-time Lattice Boltzmann Methods, one can only know the fluid state at the endpoints of the fixed time interval. During the interval, one could assume that the fluid state moves linearly through phase space in the interval, but it would be difficult to justify that assumption. A continuous-time Lattice Boltzmann Equation would (continuing with the particle-box analogy) effectively replace the strobe light with a light source that stayed on continuously throughout the duration of the experiment.

One remark regarding this metaphor. In using computer calculations, we will inevitably be constrained to work in discrete time. One might wonder then what actual benefits are obtained as a result of using a continuous-time method.

A common feature to all discrete-time Lattice Boltzmann methods is that, once the domain (the lattice) and the simulation parameters have been set, there is one and only one fixed time step that is implicitly defined by the lattice size. In other words, once the simulation has been set up, there is no way of changing the time step, and we are restricted to knowing the fluid state at discrete time points.

With continuous-time methods, the advantage is that we are not restricted to a single fixed time step. Theoretically, we can choose any positive scalar multiple of the lattice time step. Even better, we can choose any positive scalar multiple of the time step while the simulation is running. Although we are still restricted to knowing the fluid state at discrete time points, we can now control the interval of time in between these discrete time points (i.e. the time step) and we can get to a very good approximation of continuous time by defining our time step to be very small.

Thus, although we will go back to discrete time, we will gain the flexibility of being able to control the size of our time step as well as the ability to compute the state of the fluid at almost any point in time by making our time step very small. In this sense we have a 'continuous-time Lattice Boltzmann Equation'.

1.2 Outline of the paper

The first section of this paper will present the discrete-time Lattice Boltzmann Equation, using the Bhatnagar-Gross-Krook (BGK) formulation. We will present the D2Q9 model and describe its important characteristics in the second section, derive the continuous-time Lattice Boltzmann Equation from the discrete-time Lattice Boltzmann Equation for the D2Q9 model in the third section, and analyze the linear stability of global equilibria in the continuous-time Lattice Boltzmann equation in the fourth section.

The fifth section is our results section - in it, we will present a model of chaotic fluid flow in a two-dimensional domain using the continuous-time Lattice Boltzmann Equation and use the model to show that numerical implementation preserves the important characteristics of the D2Q9 model. We will also analyze some computational and numerical aspects of the continuous-time Lattice Boltzmann equation and compare these to the corresponding aspects of the discrete-time equation.

While this thesis is not intended to be a review of the extensive literature on Lattice Boltzmann Methods, we will go through some elements of the Lattice Boltzmann Method theory. More technical but nonetheless useful results from Lattice Boltzmann theory are included in the Appendices, including a brief overview of how tensor calculus is involved in defining a Lattice Boltzmann model and a derivation of the Navier-Stokes Equations from the discrete Lattice Boltzmann equation. Throughout the paper we will use the D2Q9 model, although the continuous-time Lattice Boltzmann Equation we present may be implemented for other models (including in three dimensions).

Chapter 2

The Discrete Time Lattice Boltzmann Equation

Suppose a researcher wanted to model the behavior of fluid confined to a very thin film in the shape of a square. From elementary chemistry, the film contains many particles - a single mole of water molecules would contain 10^{23} molecules. Trying to model the behavior of the fluid by modeling the behavior of the individual particles would be computationally difficult as a result of several factors:

- the number of particles that need to be tracked,
- the number of ways those particles can move within the film, and
- the number of ways those particles can interact with one another.

The Lattice Boltzmann equation changes the problem into a model that preserves many of the characteristics of the actual model but reduces the computational burden. It does this by three mechanisms:

- keeping track of the distribution of particles at a comparatively smaller number of fixed grid points instead of tracking individual particles,
- reducing the movement of particles in the region to nine different displacement vectors, and
- simplifying the collision of particles.

2.1 The D2Q9 Lattice Scheme

A Lattice Boltzmann model starts with a lattice. For convenience, we consider a regular square grid with the same number of grid points in the x- and y-directions, say $N = N_x = N_y$, so that the grid consists of N^2 identically spaced grid points. These grid points serve as reference points in the Lattice Boltzmann method, providing the method with a fixed number of locations in the region of interest at which the distribution of particles can be monitored.

To form the lattice, connect neighboring grid points with horizontal, vertical and diagonal edges, so that any interior grid point (i.e. any point not on any of the edges of the grid) is connected to every one of its neighbors by an edge. The movement of particles in Lattice Boltzmann methods is constrained to these edges. For our model, given an interior grid point at (x, y), there are nine possible displacement vectors $\boldsymbol{c}_j$ along which particles can move:

$$\begin{array}{llll} \boldsymbol{c}_0 = (0,0) & & & \\ \boldsymbol{c}_1 = (0,1) & \boldsymbol{c}_2 = (1,0) & \boldsymbol{c}_3 = (-1,0) & \boldsymbol{c}_4 = (0,-1) \\ \boldsymbol{c}_5 = (1,1) & \boldsymbol{c}_6 = (-1,1) & \boldsymbol{c}_7 = (-1,-1) & \boldsymbol{c}_8 = (1,-1) \end{array}$$

In Lattice Boltzmann Method literature, the convention is to describe the model in terms of the lattice vectors. In this model, each lattice vector is a (D=2)-tuple, and there are Q=9 lattice vectors; hence the model is known as the D2Q9 model. Associated with the lattice vectors $\boldsymbol{c}_j$ are weights, w_j:

$$w_0 = \frac{4}{9} \tag{2.1}$$

$$w_j = \frac{1}{9},\ j = 1,2,3,4 \tag{2.2}$$

$$w_j = \frac{1}{36},\ j = 5,6,7,8. \tag{2.3}$$

These weights are used because they endow the lattice with certain properties. In turn, these properties are what enable Lattice Boltzmann Methods to model fluid flow accurately with respect to the Navier Stokes Equations. For example, the weights have the property that

$$\sum_{j=0}^{8} w_j = 1 \tag{2.4}$$

$$\sum_{j=0}^{8} w_j \boldsymbol{c}_j = \boldsymbol{0}. \tag{2.5}$$

There are three additional properties that the lattice (i.e. the lattice vectors and weights) must satisfy in order for the model to display fluid behavior that agrees with the Navier-Stokes Equations:

$$\sum_{j}^{Q} w_j \boldsymbol{c}_j \boldsymbol{c}_j \quad = c_s^2 I_2 \tag{2.6}$$

$$\sum_{j}^{Q} w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \quad = 0_3 \tag{2.7}$$

$$\sum_{j}^{Q} w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \quad = \gamma c_s^4 I_4, \tag{2.8}$$

where $c_s = \frac{1}{\sqrt{3}}$, γ is a constant scalar (for the D2Q9 model, $\gamma = 1$), I_2 is the rank-2 identity tensor or the identity matrix, 0_3 is the third-rank zero tensor and I_4 is the isotropic fourth-rank identity tensor. The easiest way to understand $\boldsymbol{c}_j \boldsymbol{c}_j$ is as an outer product of each $\boldsymbol{c}_j$

with itself, and so on for outer products of more vectors. An explanation of tensors, these requirements and their significance is located in the Appendix on the tensor analysis of the D2Q9 model.

The $\boldsymbol{c}_1$-$\boldsymbol{c}_8$ correspond to the eight edges that connect the grid point at (x, y) to its nearest neighbors. These vectors fully account for the streaming or translational movement of particles along edges to adjacent locations. Particles cannot leave (x, y) in any other direction. The $\boldsymbol{c}_0$ vector accounts for particles that do not leave (x, y) i.e. particles at rest. Note that other Lattice Boltzmann methods may adopt a different numbering scheme for the displacement vectors.

Let $\boldsymbol{r}$=(x,y) be the position vector and t denote the time. Associate with each of the displacement vectors above a population function $f_j, j = 0, 1, ..., 9$ that describes in a proportional sense how much of the total particle population at (x, y) is moving along each displacement vector. Now we can define some important physical quantities:

$$\text{Mass density}, \rho(\boldsymbol{r}, t) = \sum_{j=0}^{8} f_j(\boldsymbol{r}, t) \tag{2.9}$$

$$\text{Momentum density}, \boldsymbol{\pi}(\boldsymbol{r}, t) = \sum_{j=0}^{8} f_j(\boldsymbol{r}, t)\, \boldsymbol{c}_j(\boldsymbol{r}, t) \tag{2.10}$$

$$\text{Velocity density}, \boldsymbol{u}(\boldsymbol{r}, t) = \frac{\boldsymbol{\pi}(\boldsymbol{r}, t)}{\rho(\boldsymbol{r}, t)} \tag{2.11}$$

To illustrate how particles stream from one position to another, let us momentarily ignore the possibility of collisions between particles, so that a population of particles traveling along the $\boldsymbol{c}_j$ direction would do so indefinitely. Restrict the movement of particles between grid points to discrete steps, Δt, and consider an arbitrary interior point at time t. How will the particle distributions f_j at $\boldsymbol{r}$ contribute to the neighboring grid points of $\boldsymbol{r}$ at time $t + \Delta t$? Since the particles are constrained to move on the lattice, we would expect that the particle population f_j associated with $\boldsymbol{c}_j$ at $\boldsymbol{r}$ would move along $\boldsymbol{c}_j$ into $\boldsymbol{r} + \boldsymbol{c}_j$ and contribute to the proportion f_j at $\boldsymbol{r} + \boldsymbol{c}_j$:

$$f_j(\boldsymbol{r} + \boldsymbol{c}_j, t + \Delta t) = f_j(\boldsymbol{r}, t). \tag{2.12}$$

Currently, the model accounts for particles at rest through one zero vector and for movement of particles in the domain through eight distinct displacement vectors. To complete the model, we describe how particles can interact with each other through collisions, using the Bhatnagar-Gross-Krook (BGK) approximation collision operator. Following Succi, the collision operator $C_{BGK}(f_j)$ is defined as

$$C_{\mathrm{BGK}}(f_j) = \frac{\left[f_j^{(\mathrm{eq})} - f_j\right]}{\tau}, \tag{2.13}$$

where τ is strictly positive and is often referred to as the relaxation time that is associated with relaxation of the fluid to local equilibrium due to collisions. To see what this means, notice that $C_{BGK}(f_j)$ is zero when $f_j = f_j^{(eq)}$ - that is, the equilibrium distribution function $f_j^{(eq)}$ is the distribution at which the collision operator vanishes. Thus there are no collisions if and only if the population distribution function f_j is at its equilibrium, $f_j^{(eq)}$.

What if $f_j \neq f_j^{(eq)}$? If the difference $\left[f_j^{(eq)} - f_j\right]$ is positive, then $f_j < f_j^{(eq)}$, so if we added the collision operator to the right-hand side of Eq. (2.12), the collision operator would nudge the packet of particles f_j closer to its equilibrium $f_j^{(eq)}$. Similarly, if the difference $f_j^{(eq)} - f_j$ is negative, then the collision operator is negative, and adding the collision operator to the right-hand side of Eq. (2.12) would also nudge f_j closer to $f_j^{(eq)}$.

The parameter τ magnifies the difference $\left[f_j^{(eq)} - f_j\right]$. For the same value of the difference $\left[f_j^{(eq)} - f_j\right]$, a smaller value of τ (closer to zero) leads to a larger contribution from the collision operator C_{BGK}, and a larger value of τ leads to a smaller contribution. Thus if the time scale associated with collisional relaxation to the local equilibrium $f_j^{(eq)}$ is large, then the contribution to f_j due to the collision operator is small, and thus it takes a large number of "nudges" from the collision operator for $f_j = f_j^{(eq)}$. Conversely, if the time scale is small then the contribution to f_j due to C_{BGK} is large and it takes a fewer number of adjustments from the collision operator for $f_j = f_j^{(eq)}$. This is why the parameter τ reflects the time scale required for the collision operator to adjust the f_j to their local equilibrium.

There are multiple ways of defining the equilibrium distribution function $f_j^{(eq)}$ that emphasize the importance of various parameters, but the $f_j^{(eq)}$ must derive from the f_j via the conserved quantities ρ and $\boldsymbol{\pi}$. In this paper we will use two definitions, one from Sukop and Thorne [9],

$$f_j^{\text{eq}}(\rho, \boldsymbol{u}) = \rho w_j[1 + 3\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) + 4.5\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 - 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)]. \tag{2.14}$$

and another from Boghosian [3],

$$f_j^{\text{eq}}(\rho, \boldsymbol{\pi}) = w_j \left[\rho + \frac{1}{c_s^2}\boldsymbol{\pi} \cdot \boldsymbol{c}_j + \frac{1}{2c_s^4\rho}\boldsymbol{\pi} \cdot \left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2 I_2\right) \cdot \boldsymbol{\pi}\right]. \tag{2.15}$$

First, note that these two forms are clearly interchangeable - by substituting $c_s = \frac{1}{\sqrt{3}}$, letting I_2 be the 2x2 identity matrix and pulling out a common factor of ρ outside the square brackets, one can proceed from the second form to the first form.

As to why these particular equations and why this particular notation of $c_s = \frac{1}{\sqrt{3}}$ is chosen, one answer is that it is because they are common in the Lattice Boltzmann literature and because they enable Lattice Boltzmann Methods to simulate fluid flow according to the INSE. As this paper does not attempt to review Lattice Boltzmann methods, we will give only basic explanations of why these equations are chosen.

The Incompressible Navier Stokes Equations work (as the name suggests) when a fluid is incompressible. Under many circumstances of interest to researchers (e.g. room temperature and pressure) most fluids such as air and water are incompressible. However, under extreme circumstances, fluids will become compressible; air becomes compressible at the speed of sound, for example.

We already know that the speed of the fluid can be calculated from the f_j, but we need to incorporate the fact that fluids can 'lose incompressibility' at high speeds into our model. For this reason, a 'lattice speed of sound' is incorporated into the Lattice Boltzmann Model, so as to allow for the fact that at high speeds the fluid changes properties. In the same way that the speed of light is often denoted as c, we denote this lattice speed of sound as c_s; for the D2Q9 model $c_s = \frac{1}{\sqrt{3}}$.

As for the equations for the $f_j^{(eq)}$, these are chosen so that given any f_j at $(\boldsymbol{r}, t)$, we have

$$\sum_{j=0}^{8} f_j^{(eq)} = \rho = \sum_{j=0}^{8} f_j \tag{2.16}$$

$$\sum_{j=0}^{8} f_j^{(eq)} \boldsymbol{c}_j = \boldsymbol{\pi} = \sum_{j=0}^{8} f_j \boldsymbol{c}_j. \tag{2.17}$$

$$\tag{2.18}$$

In other words, the equilibrium distribution of the fluid at a particular $(\boldsymbol{r}, t)$ conserves the mass and momentum of the fluid at $(\boldsymbol{r}, t)$.

In working with fluid, one recurrent theme is that fluids conserve mass and momentum. So far, we have shown how the lattice can be used to compute the mass and momentum of the fluid at any lattice site. However, we have not shown how the lattice can be used to describe the fluid when it is at a steady state.

The equations above give a way to compute the distributions f_j of a fluid with mass ρ and momentum $\boldsymbol{\pi}$ when it is at equilibrium, such that a fluid can transition from any state into an equilibrium state and conserve mass and momentum. Wolf-Gladrow ([10]) provides explanations of how the equations are derived.

For notational clarity we will rename the collision operator $C_{BGK}(f_j)$ as Ω_j so as to avoid any confusion with $\boldsymbol{c}_j$, the lattice vectors. Observe that Ω_j is nonlinear in the physical quantities $\boldsymbol{u}$ and ρ. In fact the collision operator is the only source of nonlinearity in the Lattice Boltzmann method. Nonlinearity is important because it makes the Lattice Boltzmann method interesting, and it is the key ingredient which allows Lattice Boltzmann methods to simulate fluid flow.

To show that the $\sum_{j=0}^{8} f_j^{(eq)} = \sum_{j=0}^{8} f_j$, we can show that sum of the collision operators will vanish for arbitrary $\boldsymbol{u}$ and ρ:

Claim:

$$\sum_{j=0}^{8} \Omega_j = \sum_{j=0}^{8} \frac{1}{\tau}(f_j^{(eq)} - f_j) = \frac{1}{\tau}\left(\sum_{j=0}^{8} f_j^{\text{eq}} - \sum_{j=0}^{8} f_j\right) = 0. \tag{2.19}$$

Proof. Let $\boldsymbol{u}$ and ρ be arbitrary. We want to show It suffices to show that

$$\begin{aligned}
\rho = \sum_{j=0}^{8} f_j &= \sum_{j=0}^{8} f_j^{(eq)} \\
&= \sum_{j=0}^{8} \rho w_j \left[1 + 3\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) + 4.5\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 - 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\right] \\
&= \rho\left(\sum_{j=0}^{8} w_j + 3\sum_{j=0}^{8} w_j\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) + 4.5\sum_{j=0}^{8} w_j\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 - 1.5\sum_{j=0}^{8} w_j\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\right).
\end{aligned} \tag{2.20}$$

Let $\boldsymbol{u} = (u_x, u_y)$. Then

$$\begin{aligned}\sum_{j=0}^{8} w_j \left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) &= \frac{1}{9}\left(u_x + u_y - u_x - u_y\right) \\ &\quad + \frac{1}{36}\left(\left(u_x + u_y\right) + \left(-u_x + u_y\right) + \left(-u_x - u_y\right) + \left(u_x - u_y\right)\right) \\ &= 0,\end{aligned} \tag{2.21}$$

and

$$\begin{aligned}4.5\sum_{j=0}^{8} w_j \left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 &= 4.5\left(\frac{1}{9}\left(2u_x^2 + 2u_y^2\right) + \frac{1}{36}\left(2\left(u_x + u_y\right)^2 + \left(2\left(u_x^2 - u_y^2\right)\right)\right)\right) \\ &= 4.5\left(\frac{1}{9}\left(2u_x^2 + 2u_y^2\right) + \frac{1}{36}\left(4u_x^2 + 4u_y^2\right)\right) \\ &= 4.5\left(\frac{1}{3}\left(u_x^2 + u_y^2\right)\right) \\ &= 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\end{aligned} \tag{2.22}$$

so

$$\begin{aligned}\rho &= \rho\left(\sum_{j=0}^{8} w_j + 3\sum_{j=0}^{8} w_j \left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) + 4.5\sum_{j=0}^{8} w_j \left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 - 1.5\sum_{j=0}^{8} w_j \left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\right) \\ &= \rho\left(\sum_{j=0}^{8} w_j + 3 \cdot 0 + 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right) - 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\right) \\ &= \rho\left(\sum_{j=0}^{8} w_j\right) \\ &= \rho \cdot 1.\end{aligned} \tag{2.23}$$

So $\sum_{j=0}^{8} f_j^{\mathrm{eq}} = \sum_{j=0}^{8} f_j$. □

We can use the properties of the lattice to show

$$\begin{aligned}\sum_{j}^{8} f_j^{(eq)} &= \rho = \sum_{j}^{8} f_j \\ \sum_{j}^{8} f_j^{(eq)} \boldsymbol{c}_j &= \pi = \sum_{\mathbf{j}}^{\mathbf{8}} \mathbf{f_j c_j}.\end{aligned}$$

Recall that

$$\begin{aligned}
\sum_j^8 w_j \boldsymbol{c}_j &= \mathbf{0} \\
\sum_j^8 w_j \boldsymbol{c}_j \boldsymbol{c}_j &= c_s^2 I_2 \\
\sum_j^8 w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j &= \mathbf{0_3} \\
\sum_j^8 w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j &= T_4.
\end{aligned}$$

Now, using Eq. (2.15), we have

$$\begin{aligned}
\sum_j^8 f_j^{(eq)} &= \rho \sum_j w_j + \frac{1}{c_s^2}\boldsymbol{\pi} \cdot \sum_j w_j \boldsymbol{c}_j + \frac{1}{2c_s^4\rho}\boldsymbol{\pi} \cdot \left[\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j - c_s^2 I_2 \sum_j w_j\right] \cdot \boldsymbol{\pi} \\
&= \rho \cdot + \frac{1}{c_s^2}\boldsymbol{\pi} \cdot 0 + \frac{1}{2c_s^4\rho}\boldsymbol{\pi} \cdot \left[c_s^2 I_2 - c_s^2 I_2\right] \cdot \boldsymbol{\pi} \\
&= \rho, \qquad (2.24)
\end{aligned}$$

and we also have

$$\begin{aligned}
\sum_j^8 f_j^{(eq)} \boldsymbol{c}_j &= \rho \sum_j w_j \boldsymbol{c}_j + \frac{1}{c_s^2}\boldsymbol{\pi} \cdot \sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j + \frac{1}{2c_s^4\rho}\boldsymbol{\pi} \cdot \left[\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j - c_s^2 \boldsymbol{c}_j \sum_j w_j\right] \cdot \boldsymbol{\pi} \\
&= \rho \cdot \mathbf{0} + \frac{1}{c_s^2}\boldsymbol{\pi}(c_s^2 I_2) + \frac{1}{2c_s^4\rho}\boldsymbol{\pi} \cdot \left[\mathbf{0_3} - c_s^2 \sum_j \boldsymbol{c}_j\right] \cdot \boldsymbol{\pi} \\
&= \mathbf{0} + \boldsymbol{\pi} + \mathbf{0} \\
&= \boldsymbol{\pi} \qquad (2.25)
\end{aligned}$$

Now we can define the discrete-time Lattice Boltzmann Equation

$$f_j\left(\boldsymbol{r} + \boldsymbol{c}_j, t + \Delta t\right) = f_j(\boldsymbol{r}, t) + \Omega_j(\boldsymbol{r}, t). \qquad (2.26)$$

The only additional term compared to Eq. (2.12) is the collision operator Ω_j. Now, the jth distribution function of particles receives a contribution from particles that stream directly into the $\boldsymbol{r}+\boldsymbol{c}_j$ position as well as a contribution from the particles that collide at $\boldsymbol{r}$ and 'bounce off' into the $\boldsymbol{r}+\boldsymbol{c}_j$ position.

2.2 Conservation Laws

In simulating fluid flow, one of the most important things to do is to verify that the equation describing fluid flow conserves mass and momentum. we do that now for the discrete-time

Lattice Boltzmann Equation in preparation for proving conservation of mass and momentum for the continuous-time Lattice Boltzmann Equation.

Define the global mass (or just simply the 'mass') of fluid on the lattice at time t as

$$M(t) = \sum_{\boldsymbol{r}} \sum_{j} f_j \tag{2.27}$$

and the global momentum of the fluid as

$$\boldsymbol{P}(t) = \sum_{\boldsymbol{r}} \sum_{j} f_j \boldsymbol{c}_j. \tag{2.28}$$

For a periodic lattice, conservation of mass is proved as follows:

$$\begin{aligned} M(t+\Delta t) &= \sum_{\boldsymbol{r}} \sum_{j} \left(f_j(\boldsymbol{r},t) + \Omega_j(\boldsymbol{r},t)\right) \\ &= \sum_{\boldsymbol{r}} \sum_{j} f_j(\boldsymbol{r},t) + \sum_{\boldsymbol{r}} \sum_{j} \Omega_j(\boldsymbol{r},t) \\ &= \sum_{\boldsymbol{r}} \sum_{j} f_j(\boldsymbol{r},t) + 0 \\ &= M(t). \end{aligned} \tag{2.29}$$

Conservation of momentum is proved similarly:

$$\begin{aligned} \boldsymbol{P}(t+\Delta t) &= \sum_{\boldsymbol{r}} \sum_{j} \left(f_j(\boldsymbol{r},t) + \Omega_j(\boldsymbol{r},t)\right) \boldsymbol{c}_j \\ &= \sum_{\boldsymbol{r}} \sum_{j} f_j(\boldsymbol{r},t)\boldsymbol{c}_j + \sum_{\boldsymbol{r}} \sum_{j} \Omega_j(\boldsymbol{r},t)\boldsymbol{c}_j \\ &= \boldsymbol{P}(t) + \sum_{\boldsymbol{r}} \left(\boldsymbol{\pi}(\boldsymbol{r},t) - \boldsymbol{\pi}(\boldsymbol{r},t)\right) \\ &= \boldsymbol{P}(t). \end{aligned} \tag{2.30}$$

Chapter 3

The Continuous Time Lattice Boltzmann Equation

3.1 Derivation

In this section, we derive the continuous-time Lattice Boltzmann Equation from the discrete-time Lattice Boltzmann Equation. The derivation borrows heavily from Boghosian et al. [3].

We first re-write Eq. (2.26) as

$$f_j(\boldsymbol{r} + \boldsymbol{c}_j, t+1) = f_j(\boldsymbol{r}, t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] \tag{3.1}$$

where we have assumed that the time-step $\Delta t = 1$. In a manner analogous to finding the time derivative of most functions, we want to find the limit of

$$\frac{f_j(\boldsymbol{r}, t+h) - f_j(\boldsymbol{r}, t)}{h} \tag{3.2}$$

as $h \to 0$.

We begin with Eq. (3.1), stepping forward in time for a proportion $h\Delta t = h$ and for a distance $h\boldsymbol{c}_j$, where $0 < h < 1$. We also weight the collision operator by h:

$$f_j(\boldsymbol{r} + h\boldsymbol{c}_j, t+h) = f_j(\boldsymbol{r}, t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right]. \tag{3.3}$$

The advantage of modifying Eq. (3.1) in this way is that if we substitute $h = 0$ or $h = 1$ we recover

$$\begin{aligned} f_j(\boldsymbol{r} + 0 \cdot \boldsymbol{c}_j, t+0) = f_j(\boldsymbol{r}, t) &= f_j(\boldsymbol{r}, t) + \frac{0}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] \\ f_j(\boldsymbol{r} + 1 \cdot \boldsymbol{c}_j, t+1) = f_j(\boldsymbol{r} + \boldsymbol{c}_j, t+1) &= f_j(\boldsymbol{r}, t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] \end{aligned}$$

so the modification does not fundamentally change the meaning of Eq. (3.1). However, if $0 < h < 1$, then the position $\boldsymbol{r} + h\boldsymbol{c}_j$ does not lie on a lattice site. We must weight Eq. (3.3) so that distributions stay on lattice sites (that is, the first argument of f_j must be of the

form $\boldsymbol{r} \pm \boldsymbol{c}_j$, since the fluid particle distributions are defined to exist only on the $\boldsymbol{r} = (x, y)$ and the only way to move from one lattice site to another is by the vectors $\boldsymbol{c}_j$). We obtain this by multiplying the right hand side of Eq. (3.3) with weight $1 - h$:

$$(1-h)\left\{f_j(\boldsymbol{r},t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]\right\}. \tag{3.4}$$

If we let $h = 0$ then (3.4) equals $f_j(\boldsymbol{r}, t)$ - i.e. if we have not stepped forward in time, then none of the fluid particles at $(\boldsymbol{r}, t)$ have moved. On the other hand, if $h = 1$ then (3.4) equals 0. This implies that (3.4) describes the proportion of the distribution $f_j(\boldsymbol{r}, t)$ that remains behind at position $\boldsymbol{r}$ after stepping forward h units of time. In the same way, multiplying the right hand side of Eq. (3.3) with weight h gives

$$h\left\{f_j(\boldsymbol{r},t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]\right\}. \tag{3.5}$$

This time, if we let $h = 1$ we recover $f_j(\boldsymbol{r} + \boldsymbol{c}_j, t + 1)$, and if we let $h = 0$ then (3.5) equals 0. Thus (3.5) describes the proportion of the distribution $f_j(\boldsymbol{r}, t)$ that moves to $\boldsymbol{r} + \boldsymbol{c}_j$ after stepping forward h units of time. Since we are interested in what happens at $\boldsymbol{r}$ and not $\boldsymbol{r} + \boldsymbol{c}_j$, we need only substitute $\boldsymbol{r}$ in (3.5) with $\boldsymbol{r} - \boldsymbol{c}_j$, so that

$$h\left\{f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)\right]\right\} \tag{3.6}$$

describes the proportion of the distribution $f_j(\boldsymbol{r} - \boldsymbol{c}_j, t)$ that moves to $\boldsymbol{r}$ after stepping forward h units of time. Since (3.4) and (3.6) describe the movement of particles at $\boldsymbol{r}$, we can add them to get the full contribution to the distribution $f_j(\boldsymbol{r}, t)$ from particles remaining at $(\boldsymbol{r}, t)$ and from particles streaming in from $\boldsymbol{r} - \boldsymbol{c}_j$:

$$\begin{aligned} f_j(\boldsymbol{r},t+h) = (1-h)&\left\{f_j(\boldsymbol{r},t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]\right\} \\ &+ h\left\{f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) + \frac{h}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)\right]\right\} \end{aligned} \tag{3.7}$$

or

$$\begin{aligned} f_j(\boldsymbol{r},t+h) = (1-h)&f_j(\boldsymbol{r},t) + hf_j(\boldsymbol{r}-\boldsymbol{c}_j,t) \\ &+ \frac{h(1-h)}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] + \frac{h^2}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)\right] \end{aligned} \tag{3.8}$$

Subtract $f_j(\boldsymbol{r}, t)$ from both sides, divide by h and take the limit as $h \to 0$ to obtain the system of differential equations

$$\frac{df_j(\boldsymbol{r},t)}{dt} = f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right], \tag{3.9}$$

which is a Lattice Boltzmann Equation that is continuous in time.

3.2 Conservation laws

Eq. (3.9) inherits all of the conservation laws of the original LBE, Eq. (3.1). Denote the global sums of mass and momentum by

$$M(t) := \sum_{\boldsymbol{r}} \rho(\boldsymbol{r},t) = \sum_{\boldsymbol{r}} \sum_{j} f_j(\boldsymbol{r},t) \tag{3.10}$$

$$\boldsymbol{P}(t) := \sum_{\boldsymbol{r}} \boldsymbol{\pi}(\boldsymbol{r},t) = \sum_{\boldsymbol{r}} \sum_{j} \boldsymbol{c}_j f_j(\boldsymbol{r},t), \tag{3.11}$$

where the spatial sums are taken over all $\boldsymbol{r}$ on the lattice. From Eq. (2.19) we know that at every lattice site $\boldsymbol{r}$, the sum over j of the equilibrium distribution is chosen to have the same value of this density as the distribution at any site. We derive the conservation of momentum,

$$\begin{aligned}
\frac{d\boldsymbol{P}}{dt} &= \sum_{\boldsymbol{r}} \sum_{j} \frac{df_j(\boldsymbol{r},t)}{dt} \boldsymbol{c}_j \\
&= \sum_{\boldsymbol{r}} \sum_{j} \left\{ f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t) \right] \right\} \boldsymbol{c}_j \\
&= \boldsymbol{P}(t) - \boldsymbol{P}(t) + \frac{1}{\tau} \sum_{\boldsymbol{r}} \left[\boldsymbol{\pi}(\boldsymbol{r},t) - \boldsymbol{\pi}(\boldsymbol{r},t) \right] \\
&= \boldsymbol{0},
\end{aligned} \tag{3.12}$$

and the conservation of mass,

$$\begin{aligned}
\frac{dM}{dt} &= \sum_{\boldsymbol{r}} \sum_{j} \frac{df_j(\boldsymbol{r},t)}{dt} \\
&= \sum_{\boldsymbol{r}} \sum_{j} \left\{ f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t) \right] \right\} \\
&= M(t) - M(t) + \frac{1}{\tau} \sum_{\boldsymbol{r}} \left[\rho(\boldsymbol{r},t) - \rho(\boldsymbol{r},t) \right] \\
&= 0,
\end{aligned} \tag{3.13}$$

where we have assumed a periodic or infinite lattice. These equations show that the rate of change of momentum and mass are both zero, so global momentum and global mass are conserved. Provided that the numerical scheme used to integrate the differential equation (i.e. the continuous-time Lattice Boltzmann Equation) forward in time preserve the linearity of the equation being integrated, then global momentum and global mass are conserved.

We require the metohd to preserve the linearity of the equation because we need to be able to pass the time derivative operator $\frac{d}{dt}$ through the sums over $\boldsymbol{r}$ and j. Without linearity, we cannot make the crucial step in equating the time derivative of global mass (for example) with a sum over $\boldsymbol{r}$ and j of the individual $\frac{df_j}{dt}$. In other words, if the numerical method used approximates $\frac{dx}{dt}$ by A and $\frac{dy}{dt}$ by B then the same method should approximate $\frac{d}{dt}(x+y)$ by $A+B$. Most reasonable solvers for ordinary differential equations respect linearity in this way.

3.3 Stability Analysis

Having proposed a new Lattice Boltzmann Equation, we must analyze its stability properties. There is at least one important reason for this, namely that we want to ensure that the method is robust with respect to small errors. If small errors grow and lead to large errors later on, this would suggest that the method is not stable in the sense that small perturbations to the distributions would lead to nonsensical output, and it would make the method unsuitable for use. If on the other hand small errors remain small (or better yet, grew smaller) then that would suggest that the method is robust with respect to small errors, and we could be more confident that we the method would not generate spurious results.

To begin understanding the stability properties of the continuous-time Lattice Boltzmann Equation, we must first understand some properties of the discrete-time Lattice Boltzmann Equation, which we reproduce here:

$$f_j\left(\boldsymbol{r}+\boldsymbol{c}_j, t+1\right) = f_j(\boldsymbol{r}, t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] \tag{3.14}$$

Note that we have replaced Δt with 1 for simplicity.

Computing the Taylor expansion of Eq. (3.14) up to first order in time and second order in $\boldsymbol{c}_j$ yields

$$f_j\left(\boldsymbol{r}+\boldsymbol{c}_j, t+1\right) = f_j, (\boldsymbol{r}, t) + 1 \cdot \frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t) \tag{3.15}$$

where we have suppressed the higher-order terms in $\boldsymbol{c}_j$. Subtracting Eq. (3.15) from Eq. (3.14) and adding $\frac{\partial f_j(\boldsymbol{r}, t)}{\partial t}$ to both sides yields

$$\frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} = -\boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) - \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] \tag{3.16}$$

We will not prove it here, but in deriving the Navier-Stokes Equations from the discrete-time Lattice Boltzmann Equation, the viscosity ν is shown to be proportional to the relaxation time τ minus the coefficient of the $\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t)$ term. This means that for the discrete-time Lattice Boltzmann Equation, the viscosity is related to the relaxation time by

$$\nu = \frac{1}{3}\left(\tau - \frac{1}{2}\right). \tag{3.17}$$

The significance of this relation is that τ can never be less than $\frac{1}{2}$ - if it were, then the viscosity would be negative, and this would lead to non-physical behavior. In fact, if τ is close to $\frac{1}{2}$, the simulation becomes unstable rapidly (depending on how close τ is to 1/2). Thus the viscosity is in a sense lower than it should be due to the extra $-\frac{1}{2}$ term.

Now we will examine the viscosity of the continuous-time Lattice Boltzmann Equation. We compute the Taylor expansion of $f_j(\boldsymbol{r}-\boldsymbol{c}_j, t)$ up to first order in time and second order in $\boldsymbol{c}_j$ to get

$$f_j(\boldsymbol{r}-\boldsymbol{c}_j, t) = f_j(\boldsymbol{r}, t) + 0 \cdot \frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} - \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) + \frac{1}{2}(-\boldsymbol{c}_j \cdot \nabla)^2 f_j(\boldsymbol{r}, t) \tag{3.18}$$

and substituting this into Eq. (3.9) gives

$$\frac{df_j(\boldsymbol{r},t)}{dt} = -\boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[{f^{(\mathrm{eq})}}_j(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right], \tag{3.19}$$

where the $-f_j(\boldsymbol{r},t)$ term in the Taylor expansion of $f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)$ has canceled the $f_j(\boldsymbol{r},t)$ term in Eq. (3.9). The only difference between Eq. (3.19) and Eq. (3.16) is in the signs on the $\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t)$ term. From Eq. (3.19) the relation is now that

$$\nu = \frac{1}{3}\left(\tau + \frac{1}{2}\right) \tag{3.20}$$

and now we see that in the continuous-time Lattice Boltzmann Equation, the effective viscosity is still not perfectly proportional - there is an extra $+\frac{1}{2}$ term in the parentheses.

Making $\tau < 0$ is not an option because this would have an undesirable effect on the collision operator: instead of constraining distributions to approach their respective local equilibrium (as is the case when $\tau > 0$), distributions would move away from their respective local equillibrium. We have effectively swapped a lower bound of $\frac{1}{2}$ with a lower bound of $-\frac{1}{2}$ for τ; we still have the problem that viscosity is not exactly proportional to τ. While the improportionality is not necessarily a problem, we can certainly do better and obtain a perfectly proportional relationship between τ and ν.

Suppose that $\tau \to \infty$ so that the collision operator vanishes and the distributions stream along their respective directions:

$$\frac{df_j(\boldsymbol{r},t)}{dt} = f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r},t). \tag{3.21}$$

If we were to describe a plane wave by

$$f_j(\boldsymbol{r},t) = e^{i\boldsymbol{k}\cdot\boldsymbol{r}+\lambda_j t}, \tag{3.22}$$

and substitute it into Eq. (3.21), we would get

$$\lambda_j e^{i\boldsymbol{k}\cdot\boldsymbol{r}+\lambda_j t} = e^{i\boldsymbol{k}\cdot(\boldsymbol{r}-\boldsymbol{c}_j)+\lambda_j t} - e^{i\boldsymbol{k}\cdot\boldsymbol{r}+\lambda_j t} \tag{3.23}$$

which simplifies to

$$\lambda_j = e^{i\boldsymbol{k}\cdot(-\boldsymbol{c}_j)} - 1. \tag{3.24}$$

Eq. (3.24) is known as a dispersion relation. Dispersion relations are commonly used in physics as a way of evaluating how the time evolution of a wave varies with the spatial oscillation. Here we have used a dispersion relation to describe what effect the streaming-only continuous Lattice Boltzmann Equation has on a plane wave.

In general the λ_j are complex-valued. If the λ_j have positive real part then this would indicate that under the effect of Eq. (3.21), plane waves grow in time; if the λ_j are pure imaginary then the plane waves oscillate in time; if the λ_j have negative real part, then the plane waves decay in time. Since Eq. (3.24) shows that the λ_j will have nonpositive real part, we know that purely streaming plane waves will not grow if streaming is described by Eq. (3.21). In some sense, this reflects the presence of the additional numerical viscosity due to the additive factor of $\frac{1}{2}$, which we now know comes from the streaming part of Eq. (3.9).

To fix this problem of additive constants in the relation between viscosity and relaxation time, we will modify the streaming part of Eq. (3.9) so that the coefficient of the second-order term in $\boldsymbol{c}_j$ vanishes. Generalizing Eq. (3.9), we have

$$\frac{df_j(\boldsymbol{r},t)}{dt} = \sum_m a_m f_j(\boldsymbol{r}+m\boldsymbol{c}_j,t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]. \tag{3.25}$$

In Eq. (3.9) $a_{-1}=1$ and $a_0=-1$. In modifying the streaming term we must ensure that conservation of mass and momentum still hold, but from our derivation of the conservation laws we know that

$$\sum_{\boldsymbol{r}}\sum_j a_m f_j(\boldsymbol{r}+m\boldsymbol{c}_j,t) = a_m \cdot M, \tag{3.26}$$

where we have removed the dependence on t to indicate that global mass is constant and where we again assume a periodic lattice. To preserve conservation of mass and momentum we demand that

$$\sum_m a_m = 0. \tag{3.27}$$

To ensure that we may derive the Navier-Stokes Equations from the modified Lattice Boltzmann Equation we also require that the coefficient of the first-order term $\boldsymbol{c}_j\cdot\nabla f_j(\boldsymbol{r},t)$ in the Taylor expansion of Eq. (3.9) remains (-1). For arbitrary m,

$$\begin{aligned} f_j(\boldsymbol{r}+m\boldsymbol{c}_j,t) &= f_j(\boldsymbol{r},t) + m\boldsymbol{c}_j\cdot\nabla f_j(\boldsymbol{r},t) + \frac{1}{2}(m\boldsymbol{c}_j\cdot\nabla)^2 f_j(\boldsymbol{r},t) \\ &= f_j(\boldsymbol{r},t) + m\boldsymbol{c}_j\cdot\nabla f_j(\boldsymbol{r},t) + \frac{1}{2}m^2(\boldsymbol{c}_j\cdot\nabla)^2 f_j(\boldsymbol{r},t) \end{aligned} \tag{3.28}$$

up to second order in $\boldsymbol{c}_j$. Substituting this into Eq. (3.25) yields

$$\frac{df_j(\boldsymbol{r},t)}{dt} = \left(\sum_m m a_m\right)\boldsymbol{c}_j\cdot\nabla f_j(\boldsymbol{r},t) + \left(\frac{1}{2}\sum_m m^2 a_m\right)(\boldsymbol{c}_j\cdot\nabla)^2 f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]. \tag{3.29}$$

To be able to recover the Navier-Stokes Equations from the Lattice Boltzmann Equation, we require that the coefficient of the first-order terms in $\boldsymbol{c}_j$ equals (-1), i.e.

$$\sum_m m a_m = -1. \tag{3.30}$$

Since the coefficient of the second-order term is

$$+\sum_m \frac{m^2}{2} a_m$$

we know that the viscosity-relaxation time relation becomes

$$\nu = \frac{1}{3}\left(\tau + \sum_m \frac{m^2}{2} a_m\right). \tag{3.31}$$

Among the combinations of m and a_m that satisfy Eq. (3.30) is the combination $a_0 = 1$ and $a_1 = -1$, which gives

$$\frac{df_j(\boldsymbol{r},t)}{dt} = f_j(\boldsymbol{r},t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t) + \frac{1}{\tau}\left[{f^{(\mathrm{eq})}}_j(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]. \tag{3.32}$$

Substituting the Taylor expansion of the $f_j(\boldsymbol{r}+m\boldsymbol{c}_j,t)$ that we derived in Eq. (3.28) for the value $m = 1$ into Eq. (3.32) gives

$$\frac{df_j(\boldsymbol{r},t)}{dt} = -\boldsymbol{c}_j\cdot\nabla f_j(\boldsymbol{r},t) - \frac{1}{2}(\boldsymbol{c}_j\cdot\nabla)^2 f_j(\boldsymbol{r},t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] \tag{3.33}$$

which is the same Taylor expansion that we derived for the discrete-time Lattice Boltzmann Equation. The significance of this is that, among the class of generalized continuous-time Lattice Boltzmann Equations, there is one that yields the same qualtitative behavior as the discrete-time Lattice Boltzmann Equation. Not surprisingly, from the particular continuous-time Lattice Boltzmann Equation in Eq. (3.32) we can derive the viscosity, $\nu = \frac{1}{3}(\tau - \frac{1}{2})$.

Given the two constraints on the a_m and m in Eqs. (3.27,3.30), we could choose the right combination so that

$$\sum_m \frac{m^2}{2} a_m = 0, \tag{3.34}$$

which we do by letting $a_{-1} = \frac{1}{2}$ and $a_{+1} = -\frac{1}{2}$. This combination of a_m and m yields

$$\boxed{\frac{df_j(\boldsymbol{r},t)}{dt} = \frac{1}{2}\left[f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t)\right] + \frac{1}{\tau}\left[{f^{(\mathrm{eq})}}_j\left(\rho(\boldsymbol{r},t),\boldsymbol{\pi}(\boldsymbol{r},t)\right) - f_j(\boldsymbol{r},t)\right].} \tag{3.35}$$

with viscosity $\nu = \frac{\tau}{3}$. We will call Eq. (3.35) the "standard continuous-Lattice Boltzmann Equation" in this paper. The effect of Eq. (3.35) on a plane wave yields the following dispersion relation for the streaming operator:

$$\lambda_j = \frac{1}{2}\left(e^{-i\boldsymbol{k}\cdot\boldsymbol{c}_j} - e^{+i\boldsymbol{k}\cdot\boldsymbol{c}_j}\right) = -i\sin\left(\boldsymbol{k}\cdot\boldsymbol{c}_j\right). \tag{3.36}$$

Since λ_j is pure imaginary, errors neither grow nor decay in time; the streaming algorithm is neutrally stable.

Chapter 4

Linear stability of global equilibria

Up to this point, we have only found the dispersion relations for the streaming part of the continuous-time Lattice Boltzmann Equations, which helped in determining the stability of the streaming algorithm. For completeness we must analyze the linear stability of the entire algorithm.

Imagine a fluid state that is at an identical equilibrium state everywhere on the domain, such that the distribution functions correspond to an equilibrium distribution. Furthermore, this equilibrium distribution depends only on a particular equilibrium momentum $\boldsymbol{\pi}$ and mass density ρ - it is independent of position $\boldsymbol{r}$ and time t. Another way of describing the fluid is that it is in a spatially homogeneous equilibrium state, $f_j^{(eq)}(\rho, \boldsymbol{\pi})$.

To analyze the stability properties of the continuous-time Lattice Boltzmann Equation, we will perturb the $f_j(\boldsymbol{r}, t)$ about the equilibrium distribution by a position- and time-dependent term $g_j(\boldsymbol{r}, t)$:

$$f_j(\boldsymbol{r}, t) = f_j^{(eq)}(\rho, \boldsymbol{\pi}) + g_j(\boldsymbol{r}, t). \tag{4.1}$$

We can then find the local changes of mass and momentum density due to the perturbation by summing over j:

$$\begin{aligned}
\rho(\boldsymbol{r}, t) &= \sum_k f_k(\boldsymbol{r}, t) \\
&= \sum_k \left(f_k^{(eq)}(\rho, \boldsymbol{\pi}) + g_k(\boldsymbol{r}, t) \right) \\
&= \sum_k f_k^{(eq)}(\rho, \boldsymbol{\pi}) + \sum_k g_k(\boldsymbol{r}, t)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\pi}(\boldsymbol{r}, t) &= \sum_k f_k(\boldsymbol{r}, t) \boldsymbol{c}_k \\
&= \sum_k \left(f_k^{(eq)}(\rho, \boldsymbol{\pi}) + g_k(\boldsymbol{r}, t) \right) \boldsymbol{c}_k \\
&= \sum_k f_k^{(eq)}(\rho, \boldsymbol{\pi}) \boldsymbol{c}_k + \sum_k g_k(\boldsymbol{r}, t) \boldsymbol{c}_k
\end{aligned}$$

so that local changes of mass and momentum density due to the perturbation are

$$\Delta\rho(\boldsymbol{r},t) = \sum_k g_k(\boldsymbol{r},t) \tag{4.2}$$

$$\Delta\boldsymbol{\pi}(\boldsymbol{r},t) = \sum_k \boldsymbol{c}_k g_k(\boldsymbol{r},t). \tag{4.3}$$

Using Eq. (2.15)

$$f_j^{\mathrm{eq}}(\rho,\boldsymbol{\pi}) = w_j\left[\rho + \frac{1}{c_s^2}\boldsymbol{\pi}\cdot\boldsymbol{c}_j + \frac{1}{2c_s^4\rho}\boldsymbol{\pi}\cdot\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2 I_2\right)\cdot\boldsymbol{\pi}\right]$$

as our working definition of the equilibrium distribution function we can apply the analog of the idea of a total derivative from elementary calculus to find the total differential of the equilibrium distribution:

$$\begin{aligned}\Delta f_j^{(eq)}(\rho,\boldsymbol{\pi}) &= \frac{\partial f_j^{(eq)}}{\partial\rho}\Delta\rho + \frac{\partial f_j^{(eq)}}{\partial\boldsymbol{\pi}}\cdot\Delta\boldsymbol{\pi} \\ &= w_j\left[1 - \frac{1}{2c_s^4\rho^2}\boldsymbol{\pi}\cdot\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2\mathbf{1}\right)\cdot\boldsymbol{\pi}\right]\Delta\rho \\ &\quad + w_j\left[\frac{\boldsymbol{c}_j}{c_s^2} + \frac{1}{c_s^4\rho}\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2\mathbf{1}\right)\cdot\boldsymbol{\pi}\right]\cdot\Delta\boldsymbol{\pi}\end{aligned} \tag{4.4}$$

We can now substitute Eqs. (4.3,4.3) into Eq. (4.4), pull out a factor of $g_j(\boldsymbol{r},t)$ and collect the sum over k:

$$\begin{aligned}\Delta f_j^{(eq)}(\rho,\boldsymbol{\pi}) &= \sum_k g_k(\boldsymbol{r},t)w_j\left[1 - \frac{1}{2c_s^4\rho}\boldsymbol{\pi}\cdot\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2\mathbf{1}\right)\cdot\boldsymbol{\pi} + \frac{\boldsymbol{c}_j\cdot\boldsymbol{c}_k}{c_s^2} + \frac{1}{c_s^4\rho}\boldsymbol{c}_k\cdot\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2\mathbf{1}\right)\cdot\boldsymbol{\pi}\right] \\ &= \sum_k g_k(\boldsymbol{r},t)w_j\left[1 + \frac{\boldsymbol{c}_j\cdot\boldsymbol{c}_k}{c_s^2} + \left(\frac{1}{c_s^4\rho}\boldsymbol{c}_k - \frac{1}{2c_s^4\rho}\boldsymbol{\pi}\right)\cdot\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2\mathbf{1}\right)\cdot\boldsymbol{\pi}\right] \\ &= \sum_k g_k(\boldsymbol{r},t)w_j\left[1 + \frac{\boldsymbol{c}_j\cdot\boldsymbol{c}_k}{c_s^2} + \frac{1}{c_s^2}\left(\boldsymbol{c}_k - \frac{\boldsymbol{\pi}}{2\rho}\right)\cdot\left(\frac{\boldsymbol{c}_j\boldsymbol{c}_j}{c_s^2} - \mathbf{1}\right)\cdot\boldsymbol{\pi}\right] \\ &= \sum_k F_{jk}g_k(\boldsymbol{r},t)\end{aligned} \tag{4.5}$$

where we have defined the matrix

$$F_{jk} := w_j\left[1 + \frac{\boldsymbol{c}_j\cdot\boldsymbol{c}_k}{c_s^2} + \frac{1}{c_s^2}\left(\boldsymbol{c}_k - \frac{\boldsymbol{\pi}}{2\rho}\right)\cdot\left(\frac{\boldsymbol{c}_j\boldsymbol{c}_j}{c_s^2} - \mathbf{1}\right)\cdot\frac{\boldsymbol{\pi}}{\rho}\right]. \tag{4.6}$$

The $f_j(\boldsymbol{r},t)$ we obtained from perturbing the spatially homogenous equilibrium distributions can then be plugged into the system of differential equations in Eq. (3.35):

$$\begin{aligned}\frac{df_j(\boldsymbol{r},t)}{dt} &= \frac{d}{dt}\left[f_j^{(eq)}(\rho,\boldsymbol{\pi}) + g_j(\boldsymbol{r},t)\right] \\ &= 0 + \frac{dg_j(\boldsymbol{r},t)}{dt}\end{aligned}$$

where we have used the fact that the spatially homogeneous equilibrium state is independent of time t and depends only on ρ and $\boldsymbol{\pi}$.

Now we want to use the generalized definition of the continuous-time Lattice Boltzmann Equation here, but we must take care to not simply replace $f_j(\boldsymbol{r}, t)$ in the original definition with $g_j(\boldsymbol{r}, t)$ because the ordinary differential equation described by the Lattice Boltzmann Equation has a nonlinear component in the collision term. We must approximate the effect of the perturbation in the collision term using Eq. (4.5), so that the linearized evolution of the perturbation becomes

$$\begin{aligned}\frac{dg_j(\boldsymbol{r}, t)}{dt} &= \sum_m a_m g_j(\boldsymbol{r} + m\boldsymbol{c}_j, t) + \frac{1}{\tau}\left(\sum_k F_{jk} g_k(\boldsymbol{r}, t) - g_j(\boldsymbol{r}, t)\right) \\ &= \sum_m a_m g_j(\boldsymbol{r} + m\boldsymbol{c}_j, t) + \frac{1}{\tau}\sum_k \left(F_{jk} - \delta_{jk}\right) g_k(\boldsymbol{r}, t),\end{aligned}$$

where δ_{jk} is the Kronecker delta - $\delta_{jk} = 1$ if $j = k$ and is zero otherwise. We want to find the dispersion relation - to express λ in terms of the 2-vectors $\boldsymbol{k}$ and $\boldsymbol{c}_j$, and we will arrive at a concise way of recording the dispersion relations in terms of matrices. Note that we no longer index the λ by j. This is because the λ will no longer be independent for each of the Q directions; instead they will emerge as the Q roots of a Q-th degree polynomial.

Suppose the perturbation can be expressed as a plane wave

$$g_j(\boldsymbol{r}, t) = \sum_{\boldsymbol{k}} A_j(\boldsymbol{k}) e^{i\boldsymbol{k}\cdot\boldsymbol{r} + \lambda t}$$

where we have chosen to represent $g_j(\boldsymbol{r}, t)$ in terms of its Fourier modes. For simplicity however, we focus only on one of the A_j (and hence one mode) in the calculations that follow, say

$$g_j(\boldsymbol{r}, t) = A_j(\boldsymbol{k}_0) e^{i\boldsymbol{k}_0\cdot\boldsymbol{r} + \lambda t} = A_j e^{i\boldsymbol{k}_0\cdot\boldsymbol{r} + \lambda t}$$

We also have chosen the continuous-time Lattice Boltzmann Equation with weights $a_{+1} = -\frac{1}{2}$ and $a_{-1} = \frac{1}{2}$ as our main example, although the calculations can be repeated for the general case.

By plugging in $g_j(\boldsymbol{r}, t)$ into the continuous-time Lattice Boltzmann Equation we obtain

$$\lambda g_j(\boldsymbol{r}, t) = \frac{1}{2}\left[g_j(\boldsymbol{r} - \boldsymbol{c}_j, t) - g_j(\boldsymbol{r} + \boldsymbol{c}_j, t)\right] + \frac{1}{\tau}\sum_\ell \left(F_{j\ell} - \delta_{j\ell}\right) g_\ell(\boldsymbol{r}, t).$$

Substitute in the plane-wave definition of $g_j(\boldsymbol{r}, t)$:

$$\lambda A_j e^{i\boldsymbol{k}\cdot\boldsymbol{r} + \lambda t} = A_j \frac{1}{2}\left[e^{i\boldsymbol{k}_0\cdot(\boldsymbol{r} - \boldsymbol{c}_j) + \lambda t} - e^{i\boldsymbol{k}_0\cdot(\boldsymbol{r} + \boldsymbol{c}_j) + \lambda t}\right] + \frac{1}{\tau}\sum_\ell \left(F_{j\ell} - \delta_{j\ell}\right) A_\ell e^{i\boldsymbol{k}\cdot\boldsymbol{r} + \lambda t}. \quad (4.7)$$

We can divide both sides of the equation by common terms to obtain

$$\begin{aligned}A_j\lambda &= A_j\frac{1}{2}\left[e^{-i\boldsymbol{k}_0\cdot\boldsymbol{c}_j} - e^{+i\boldsymbol{k}_0\cdot\boldsymbol{c}_j}\right] + \frac{1}{\tau}\sum_\ell \left(F_{j\ell} - \delta_{j\ell}\right) A_\ell \\ &= -A_j\cdot\left[i\sin(\boldsymbol{k}_0\cdot\boldsymbol{c}_j)\right] + \frac{1}{\tau}\sum_\ell \left(F_{j\ell} - \delta_{j\ell}\right) A_\ell.\end{aligned}$$

While the above result is true, it could be made more concise by judicious use of our friendly neighborhood Kronecker delta, $\delta_{j\ell}$. Notice that $A_j\lambda = A_\ell\lambda\delta_{j\ell}$, since when $\delta_{j\ell} = 1$, $\ell = j$, so we retrieve $A_j\lambda$. Using this nifty trick, we rewrite the above result in such a way that will motivate a neat summary of our dispersion relations:

$$\begin{aligned} 0 &= A_j\lambda + A_j i\sin(\boldsymbol{k}_0\cdot\boldsymbol{c}_j) - \frac{1}{\tau}\sum_\ell (F_{j\ell}-\delta_{j\ell})\,A_\ell \\ &= A_\ell\lambda\delta_{j\ell} + A_\ell i\sin(\boldsymbol{k}_0\cdot\boldsymbol{c}_j)\delta_{j\ell} - \frac{1}{\tau}\sum_\ell (F_{j\ell}-\delta_{j\ell})\,A_\ell \\ &= \left[\lambda\delta_{j\ell} + i\sin(\boldsymbol{k}_0\cdot\boldsymbol{c}_j)\delta_{j\ell} - \frac{1}{\tau}\sum_\ell (F_{j\ell}-\delta_{j\ell})\right] A_\ell. \end{aligned}$$

Recall that the indices ℓ and j range over the number of distributions (equivalently, the number of displacement vectors on the lattice). In the quantity in the square brackets we have both indices appearing, whereas there is only one index on A_ℓ. For the D2Q9 model, this suggests that A_ℓ represents a length-9 vector and the quantity in square brackets is a 9x9 square matrix, each of whose entries can be described by

$$\Lambda_{j\ell} = \lambda_j\delta_{j\ell} + i\sin(\boldsymbol{k}_0\cdot\boldsymbol{c}_j) - \frac{1}{\tau}\sum_\ell (F_{j\ell}-\delta_{j\ell})\,. \tag{4.8}$$

Now we see that the A_ℓ is a length-9 vector that contains the information about the amplitude of $g_\ell(\boldsymbol{r},t)$ in Fourier space. For a nontrivial perturbation $g_\ell(\boldsymbol{r},t)$, we must have at least one nonzero amplitude - if all the amplitudes $A_\ell = 0$ then clearly all the $g_\ell(\boldsymbol{r},t) = 0$, and our perturbation would be much less useful. The length-9 vector A_ℓ cannot be the zero vector. Then

$$\Lambda_{j\ell}A_\ell = 0 \tag{4.9}$$

implies that the matrix $\Lambda_{j\ell}$ has a nontrivial nullspace, and by elementary linear algebra

$$\det\Lambda_{j\ell} = 0. \tag{4.10}$$

We will use this to complete our linear stability analysis of global equilibria.

Our analysis above focused on the continuous-time Lattice Boltzmann Equation in Eq. (3.35), but the analysis may be adapted to the case with general weights to give

$$0 = \lambda - \left[\sum_m a_m e^{+im\boldsymbol{k}_0\cdot\boldsymbol{c}_j} + \frac{1}{\tau}\sum_\ell (F_{j\ell}-\delta_{j\ell})\,A_\ell\right]. \tag{4.11}$$

Using the D2Q9 model and Eq. (4.8), the $\boldsymbol{k}_0 = \boldsymbol{0}$ mode yields

$$\det\Lambda_{j\ell} = \lambda^3\left(\lambda + \frac{1}{\tau}\right)^6. \tag{4.12}$$

Setting $\det\Lambda_{j\ell} = 0$, we see that there are three conserved, hydrodynamic modes with $\lambda = 0$ and six with $\lambda = -1/\tau < 0$. The negative hydrodynamic modes are stable, so we analyze the stability of modes corresponding to $\boldsymbol{k}_0 \neq 0$.

Computing $\det \Lambda$ leads to a complicated sum of many terms. Using Mathematica, we found that the most unstable modes have

$$\begin{aligned} \boldsymbol{k}_0 &= k\boldsymbol{e}_x \\ \boldsymbol{\pi}/\rho = \boldsymbol{u} &= u\boldsymbol{e}_x. \end{aligned}$$

Note that $\boldsymbol{\pi}/\rho$ was irrelevant in the $\boldsymbol{k}_0 = 0$ mode but has become relevant for all other modes. In the $\boldsymbol{k}_0 = 0$ mode, the matrix $F_{j\ell}$ did not contribute in the expression for $\det \Lambda_{j\ell}$, but for all other modes $F_{j\ell}$ does contribute, and $\boldsymbol{\pi}$ and ρ are the only variable (i.e. non-lattice determined) quantities in $F_{j\ell}$. We take the ratio $\boldsymbol{\pi}/\rho$ because it reduces the number of variables that control for instability to one, namely $\boldsymbol{u}$. By the benevolence of mathematics, this vector parameter reduces further to a scalar - the velocity magnitude u.

For the $\boldsymbol{k}_0 \neq 0$ modes, we find that

$$\begin{aligned} \det \Lambda = &\left(\lambda + \frac{1}{\tau}\right)\left[\left(\lambda + \frac{1}{\tau}\right)^2 + \sin^2 k\right] \\ &\left[\lambda^3 + \frac{1}{\tau}\lambda^2 + \left(\sin^2 k + \frac{2iu\sin k}{\tau}\right)\lambda + \frac{1-3u^2}{3\tau}\sin^2 k\right] \\ &\left[\lambda\left(\lambda + \frac{1}{\tau}\right)^2 + \left(\sin^2 k + \frac{iu\sin k}{\tau}\right)\lambda + \frac{1}{3\tau}\left(\sin^2 k + \frac{3iu\sin k}{\tau}\right)\right]. \end{aligned} \tag{4.13}$$

The terms $\left(\lambda + \frac{1}{\tau}\right)$ and $\left[\left(\lambda + \frac{1}{\tau}\right)^2 + \sin^2 k\right]$ give the dispersion relations

$$\lambda_1 = -\frac{1}{\tau} \tag{4.14}$$

$$\lambda_2 = -\frac{1}{\tau} + i\sin k \tag{4.15}$$

$$\lambda_3 = -\frac{1}{\tau} - i\sin k, \tag{4.16}$$

and these all have negative real part.

The other two factors in the expression for $\det \Lambda$ must give six additional dispersion relations. We found that the first of these dispersion relations to cause instability comes from the third factor. We found the velocity at which they first destabilized by changing λ to $i\lambda$ in the third factor, setting the real and imaginary parts of the resulting expression to zero and solving for λ and u. This yielded six pairs of combinations (λ, u): $\left(0, \pm\sqrt{3}/3\right)$, $\left(+i\sin k, -1 \pm \sqrt{3}/3\right)$ and $\left(-i\sin k, +1 \pm \sqrt{3}/3\right)$. Among the combinations with pure imaginary λ, the minimum velocity required for *instability* of global equilibria is $u_{crit} = +1 - \sqrt{3}/3$. Thus linear stability of the global equilibrium is attained if

$$|\boldsymbol{u}| < 1 - \frac{\sqrt{3}}{3} \approx 0.42265\ldots \tag{4.17}$$

We can translate this to an upper bound on the Mach number by using the D2Q9 sound speed $c_s = 1/\sqrt{3}$:

$$M := \frac{|\boldsymbol{u}|}{c_s} < \sqrt{3} - 1 \approx 0.73205\ldots. \tag{4.18}$$

We sought the combinations with pure imaginary λ because we know that dispersion relations with positive real λ will yield instability and those with negative real λ will yield stability regardless of the magnitude of velocity. However, with pure imaginary λ the magnitude of the velocity becomes important because it is the other variable to appear in $\det \Lambda$; hence it becomes a parameter that controls the stability of the fluid flow.

The upper bound on the Mach number M is much higher than we would reasonably use. At sufficiently high velocities, incompressible fluids will become compressible, rendering the INSE invalid for modeling fluid flow. We wish to restrict the behavior of fluids to speeds where the fluid is firmly in the incompressible regime, so the Mach number should be kept lower than the upper bound.

The preceding analysis assumed a global equilibrium in which the fluid distributions are identical at every lattice site (i.e. the f_j are independent of both position $\boldsymbol{r}$ and time t). If spatial gradients were present in the fluid (i.e. if the f_j were independent of time t but not of position $\boldsymbol{r}$) then the upper bound on the Mach number should be lower.

Chapter 5

Numerical Results from a Model of Fluid Flow

We implemented Eq. (3.35),

$$\frac{df_j(\boldsymbol{r},t)}{dt} = \frac{1}{2}\left[f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t)\right] + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]$$

in the C programming language. The grid size used was 100 by 100 throughout, although the program can be modified to work on larger grid sizes by changing the grid size parameters. In its simplest form, the program defines a system of differential equations at each lattice site, effectively forming a 3-dimensional 100 by 100 by 9-size array. The current version of the program forms a 3-dimensional 100 by 100 by 15 array so that we can keep track of quantities in addition to the f_j at each lattice site. These quantities include the x- and y-components of hydrodynamic velocity, mass density, vorticity and the x- and y-derivatives of vorticity.

We want to allow for a force to be applied to the fluid at each lattice site so we modify Eq. (3.35) by including a force term in the equation:

$$\frac{df_j(\boldsymbol{r},t)}{dt} = \frac{1}{2}\left[f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t)\right] + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] + \frac{1}{6}\boldsymbol{c}_j \cdot \boldsymbol{R} \tag{5.1}$$

where the vector $\boldsymbol{R} = \boldsymbol{R}(\boldsymbol{r},t)$ is the force vector applied at $\boldsymbol{r}$ and time t. We chose the constant $1/6$ to modify the forcing term so that the sum

$$\sum_{\boldsymbol{r}} \frac{1}{6}\boldsymbol{c}_j\boldsymbol{c}_j = \frac{1}{6}\sum_{\boldsymbol{r}} \boldsymbol{c}_j\boldsymbol{c}_j = \mathbf{1}, \tag{5.2}$$

which then allows us to retrieve the force vector itself at $\boldsymbol{r}$:

$$\sum_{j} \frac{1}{6}\boldsymbol{c}_j \cdot \boldsymbol{R}\boldsymbol{c}_j = \frac{1}{6}\sum_{j} \boldsymbol{c}_j\boldsymbol{c}_j \cdot \boldsymbol{R} = \mathbf{1}\cdot\boldsymbol{R} = \boldsymbol{R}. \tag{5.3}$$

The conservation laws for mass and momentum describe the additional requirement on the

force field $\boldsymbol{R}$:

$$\begin{aligned}
\frac{dM}{dt} &= \sum_{\boldsymbol{r}}\sum_{j}\frac{df_j(\boldsymbol{r},t)}{dt} \\
&= \sum_{\boldsymbol{r}}\sum_{j}\left\{f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)-f_j(\boldsymbol{r},t)+\frac{1}{\tau}\left[f_J^{(eq)}(\boldsymbol{r},t)-f_j(\boldsymbol{r},t)\right]+\frac{1}{6}\boldsymbol{c}_j\cdot\boldsymbol{R}\left(\boldsymbol{r},t\right)\right\} \\
&= M(t)-M(t)+\sum_{\boldsymbol{r}}\sum_{j}\frac{1}{\tau}\left[\rho(\boldsymbol{r},t)-\rho(\boldsymbol{r},t)\right]+\frac{1}{6}\sum_{\boldsymbol{r}}\sum_{j}\boldsymbol{c}_j\cdot\boldsymbol{R}\left(\boldsymbol{r},t\right) \\
&= 0+0+\frac{1}{6}\sum_{\boldsymbol{r}}\sum_{j}\boldsymbol{c}_j\cdot\boldsymbol{R}\left(\boldsymbol{r},t\right) \qquad (5.4)
\end{aligned}$$

and

$$\begin{aligned}
\frac{d\boldsymbol{P}}{dt} &= \sum_{\boldsymbol{r}}\sum_{j}\frac{df_j(\boldsymbol{r},t)}{dt}\boldsymbol{c}_j \\
&= \sum_{\boldsymbol{r}}\sum_{j}\left\{f_j(\boldsymbol{r}-\boldsymbol{c}_j,t)-f_j(\boldsymbol{r},t)+\frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t)-f_j(\boldsymbol{r},t)\right]+\frac{1}{6}\boldsymbol{c}_j\cdot\boldsymbol{R}\left(\boldsymbol{r},t\right)\right\}\boldsymbol{c}_j \\
&= \boldsymbol{P}(t)-\boldsymbol{P}(t)+\sum_{\boldsymbol{r}}\frac{1}{\tau}\left[\boldsymbol{\pi}(\boldsymbol{r},t)-\boldsymbol{\pi}(\boldsymbol{r},t)\right]+\sum_{\boldsymbol{r}}\frac{1}{6}\sum_{j}\boldsymbol{c}_j\boldsymbol{c}_j\cdot\boldsymbol{R}\left(\boldsymbol{r},t\right) \\
&= 0+0+\sum_{\boldsymbol{r}}\boldsymbol{R}\left(\boldsymbol{r},t\right), \qquad (5.5)
\end{aligned}$$

where we have used Eq.(5.3) to remove the sum over j of the outer product $\boldsymbol{c}_j\boldsymbol{c}_j$. For conservation of mass we need $\sum_{\boldsymbol{r}}\sum_j \boldsymbol{c}_j\cdot\boldsymbol{R}(\boldsymbol{r},t)=0$ and for conservation of momentum we need $\sum_{\boldsymbol{r}}\boldsymbol{R}(\boldsymbol{r},t)=0$.

The former requirement automatically follows as a result of the lattice - any Lattice Boltzmann model has the property that the sum over j and $\boldsymbol{r}$ of the scalar product of the $\boldsymbol{c}_j\cdot\mathbf{v}$ vanishes, for any vector $\mathbf{v}$. This is because every contribution due to the scalar product $\boldsymbol{c}_j\cdot\boldsymbol{R}(\boldsymbol{r},t)$ will be negated by the scalar product corresponding to the scalar product of $\boldsymbol{R}$ with $-\boldsymbol{c}_j$; the additive inverse of every lattice vector is also a lattice vector.

The second requirement effectively ties conservation of momentum to a force field whose integral over the domain vanishes. On a discretized domain such as a lattice, taking the integral of the force field corresponds to evaluating the sum of the forcing terms over all the lattice sites. Thus if the sum of the components of the force field over the entire domain vanishes, we should expect momentum to be conserved up to floating point accuracy.

For the model we implemented, we chose a force field of the form

$$\begin{aligned}
\boldsymbol{R}(\boldsymbol{r},t) &= \left(R_x((x,y),t),R_y((x,y),t)\right. \\
&= R_0\left(\sin\left(\frac{2\pi px}{N_x}\right)\sin\left(\frac{2\pi qy}{N_y}\right),\cos\left(\frac{2\pi px}{N_x}\right)\cos\left(\frac{2\pi qy}{N_y}\right)\right) \qquad (5.6)
\end{aligned}$$

where R_0 is the magnitude of the forcing, $p,q\in\mathbb{N}$, x and y are the x- and y-coordinates of the lattice site, and N_x and N_y are the number of grid points in the x-direction and in the y-direction. There is no particularly important reason for choosing this forcing term.

With the array of differential equations established, we step forward in time using a numerical integration scheme. Two schemes that have been successfully used in our simulations are the second-order and fourth-order Runge-Kutta methods. These methods involve computing a weighted average of rate-of-change arrays, each of which is indirectly computed from the primary array using Eq. (3.35) and linear operations of scalar multiplication and addition on the vector space of three-dimensional arrays.

From the conservation law for mass density, we know that summing the entries in the rate-of-change array over all positions $\boldsymbol{r}$ should equal zero. However, we certainly hope that summing the entries in the primary array (i.e. the array containing the distributions) doesn't equal zero, even after adding the weighted average-array. We will show that the process of numerically integrating the primary array forward in time does indeed preserve the global mass of the fluid in the primary array, up to round-off error.

At this point, we have described everything that is needed to implement Eq. (3.35); we now need to be able to extract data from a simulation. The fundamental data that underpin the calculation of useful quantities such as mean energy and mean enstrophy over the domain is the velocity field on the domain.

An advantage of using the continuous-time Lattice Boltzmann Equation is that we can integrate the differential equations forward to any future point in real-valued (as opposed to integer-valued) time and compute the velocity field - something that would have been impossible with the discrete-time Lattice Boltzmann Equation, where we were constrained to integer time-steps.

In our implementation, we also included functions that would provide snapshots of various scalar and vector fields in the form of a .txt file. We wrote a separate Mathematica program to transform the .txt file into .jpeg files. Using standard .jpeg to .mpeg file conversion software such as mplayer or ffmpeg, we can compile snapshots into a movie.

5.1 Conservation of Mass and Momentum

Individual lattice sites

Analysis of the conservation of mass and momentum begins with a single lattice site. In computing the right hand side of Eq. (3.35) for a single lattice site with nine distributions, the component that is most likely to cause problems with respect to mass and momentum conservation is the computation of the equilibrium distribution function, $f_j^{(eq)}$. This is because the equilibrium distribution function is nonlinear - it has a quadratic term in $(\boldsymbol{c}_j \cdot \boldsymbol{u})$.

To see how serious the roundoff error could be, we wrote a test program that generated a random scalar and a random vector. These were supplied as a hypothetical mass density ρ and velocity $\boldsymbol{u}$ into the function that computes the equilibrium distributions from ρ and $\boldsymbol{u}$ using the definition of the equilibrium function in Eq. (2.14). We tested this function by requiring that it calculate the difference between the original density and the sum of the equilibrium distributions.

From Eq. (2.16) we know that summing the $f_j^{(eq)}$ over j should yield the density ρ. It also follows from Eq. (2.21) and Eq. (2.22) that

- the sum over j of the first term should equal one: $\sum_j \rho w_j = 1$

- the sum of the second term should equal zero: $\sum_j \rho w_j 3(\boldsymbol{c}_j \cdot \boldsymbol{u}) = 0$
- and the sum of the third term cancels out the last term: $\sum_j \rho w_j [4.5(\boldsymbol{c}_j \cdot \boldsymbol{u})^2 - 1.5(\boldsymbol{u} \cdot \boldsymbol{u})] = 0$

Our test program showed that for any reasonable random pair of a velocity vector and density scalar, the difference between the density scalar and the sum of the equilibrium distributions was on the order of 10^{-16} for a single lattice site. A discrepancy on the order of 10^{-16} is reassuring, because it means that the computation of the equilibrium distribution function does not contribute any additional discrepancy to the mass at a single lattice site above what is expected from numerical roundoff error for double-precision floating-point numbers.

In this way we verified that at individual lattice sites, the function for computing the equilibrium distributions conserved mass. Conservation of momentum at individual lattice sites follows similiarly.

Entire arrays

We found in the previous section that the average discrepancy in density due to computing the equilibrium distributions was $O(10^{-16})$. We might expect that on a 100 by 100 grid consisting of 10000 points, the discrepancy in global mass due to numerical roundoff after one numerical integration would be on the order of 10000 times 10^{-16} or $O(10^{-12})$.

To verify this, we wrote a test program that performed one numerical integration step by computing the rate-of-change array and adding it to the main array containing the fluid particle distributions, much in the same way that one might define $y_{n+1} = y_n + \Delta y_n$ as a starting approximation. The program would calculate the global mass and momentum of both the main array, y_n, and the rate-of-change array Δy_n before and after the numerical integration step. It would also calculate the difference in the global mass of the main array.

Note that unless otherwise specified, we set the initialization parameters for every test program to a standard parameter set:

- the density at each lattice site $\rho = 1.0$;
- the hydrodynamic velocity at each lattice site $\boldsymbol{u} = \boldsymbol{0}$;
- the force magnitude $R_0 = 0.0001$;
- the relaxation time parameter $\tau = 0.1$.

There were two functions that we used to compute the global mass. We found that with one function, the discrepancy in mass due to the addition of the rate-of-change array was zero. With the other we found a discrepancy on the order of 10^{-9}. The discrepancies in the x- and y-components of global momentum were calculated and shown to be on the order of 10^{-17}. That is, for global momentum, we found that the discrepancy was on the order of numerical roundoff error. This suggests that our implementation of the continuous-time Lattice Boltzmann Equation does indeed preserve momentum.

The discrepancy between the two methods used to compute the global mass arose as a result of differences in how global mass is computed. In the first case (in which there

were zero discrepancies in mass), global mass was computed on a lattice-site basis. The mass density at each lattice was calculated and stored into a variable, whose value was subsequently added to the value of the global sum. In other words we computed the global mass on a lattice-site basis. In the second function (where the discrepancies in mass were $O\left(10^{-9}\right)$, global mass was computed by adding each distribution to the global sum, or on a distribution basis.

From the different methods, we can see that computing the global mass on a distribution basis introduces more numerical errors into the computation of mass. The value of each distribution f_j is small - every distribution should be $O(\rho)$, where ρ is the average mass density assigned to a lattice site. As the second function sums over more lattice sites the running total gets larger, while the distributions remain small. By virtue of the way floating-point numbers are stored in computer arithmetic, adding a small number to an increasingly large number will introduce roundoff errors. Thus, the larger the running sum of the f_j, the more roundoff error is introduced when another f_j is added to the running total. It is not surprising therefore that this method of computing the global mass results in a discrepancy that exceeds the predicted discrepancy of $O(10^{-12})$.

One useful conclusion is that so far, our implementation of the continuous-time Lattice Boltzmann Equation agrees with the theoretical analysis in that mass and momentum are conserved. The rate-of-change array does not add or subtract a significant amount of mass or momentum to the main array. Another useful conclusion from this results analysis is that the computation of global quantities may be improved by introducing intermediary storage variables to reduce roundoff error. We adopted this strategy for computing global quantities whenever possible.

Runge-Kutta Methods

In the previous section we examined what happened to mass and momentum conservation when we used a primitive numerical integration method, whereby we computed a rate-of-change array and added it to the fluid distribution array much in the same way that one would form $y_{n+1} = y_n + \Delta y_n$. We now turn our attention to a more refined integration scheme, namely the fourth-order Runge-Kutta methods. We used the fourth-order Runge-Kutta method because this method is well-known and simple to use. In Atkinson [1] the Runge-Kutta methods have the general form

$$y_{n+1} = y_n + hF\left(x_n, y_n, h; f\right),$$

where $n \geq 0$ and we have the differential equation

$$y' = \frac{dy}{dx} = f(x, y),$$

and the initial condition

$$y\left(x_0\right) = Y_0.$$

y_n denotes the vector at time n of dependent variables, $f\left(x, y\right)$ is the differential equation describing the evolution of y_n and x_n denotes the vector of independent variables at time n.

The fourth-order Runge Kutta method is

$$y_{n+1} = y_n + \frac{h}{6}\left[V_1 + 2V_2 + 2V_3 + V_4\right]$$

where

$$\begin{aligned} V_1 &= f\left(x_n, y_n\right) \\ V_2 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hV_1\right) \\ V_3 &= f\left(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hV_2\right) \\ V_4 &= f\left(x_n + h, y_n + hV_3\right) \end{aligned}$$

To examine the effect of using a Runge-Kutta numerical integration method on the conservation of mass, we ran a test program that used the fourth-order Runge-Kutta method and computed the global mass of the main array of distributions before and after a single numerical integration step. The program also computed the global mass of the rate-of-change arrays V_1-V_4.

We first ran the test program using the standard set of parameters, and the difference in global mass for the main array of distributions was zero; the RK method conserved mass. The rate-of-change arrays were not all uniformly zero. The global mass of the first and second rate-of-change arrays were $O(10^{-19})$ and $O(10^{-15})$. The global mass was $O(10^{-13})$ for both the third and fourth arrays. The mass of the first array is very small, and numerically indistinguishable from zero since it is $O(10^{-16})$. The mass of the other arrays are larger than zero, so we cannot disregard them.

We will later show that our implementation of the continuous-time method does not perfectly conserve mass, and this is due to the discrepancies in mass for the rate-of-change arrays. We ran the test program with different parameters and observed that as the relaxation time τ decreased (i.e. viscosity ν decreased) and the magnitude of the forcing term increased, the discrepancies in the global mass of the arrays also increased.

Denote the Reynolds number as Re:

$$Re = \frac{UL}{\nu} \tag{5.7}$$

where U is often known as the characteristic velocity of the flow, L is the characteristic length of the domain (in our case, simply the length of the domain itself) and ν is hydrodynamic viscosity. The velocity of the flow is directly related to the force applied to the fluid, as higher forcing terms will lead to higher velocities. The Reynolds number is a dimensionless parameter that is used by fluid dynamicists to control whether a fluid is turbulent - higher Reynolds numbers lead to more turbulent flow, and lower Reynolds numbers lead to laminar flows.

If decreasing viscosity and increasing the forcing term leads to larger discrepancies, then higher Reynolds numbers for a fluid flow simulation will exhibit increasing mass discrepancies. This is a problem when one wishes to simulate high-Reynolds number flow. We will leave solution of this problem for future work.

5.2 Validity Testing

The main motivation for implementing Lattice-Boltzmann Methods is to simulate the behavior of incompressible fluids. In fluid dynamics theory one often uses the Incompressible Navier-Stokes Equations to describe fluid behavior:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nu \nabla^2 \boldsymbol{u} \tag{5.8}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{5.9}$$

where the ∇ operator is $< \frac{\partial}{\partial x}, \frac{\partial}{\partial y} >$, $\boldsymbol{u}$ is the velocity of the fluid ν is the viscosity and p is the pressure. One way that we can test the validity of a method for simulating fluid behavior is to use properties about the velocity vector field that one "inherits" from the Navier-Stokes Relations and see whether the simulated fluid satisfies these properties. One property that can be derived from the Navier-Stokes Equations is that the rate of change of the mean energy in the domain is proportional to the mean enstrophy in the domain. Frisch [4] calls this property the 'energy balance equation', and we will derive the energy balance equation here.

Eqs. (5.8) and (5.9) can be rewritten as

$$\partial_t u_i + u_j \partial_j u_i = -\partial_i p + \nu \partial_{jj} u_i \tag{5.10}$$

and

$$\partial_i u_i = 0 \tag{5.11}$$

where we adopt the convention that a repeated index on both sides of the equation indicates summation and

$$\partial_i = \frac{\partial}{\partial x_i} \tag{5.12}$$

$$\partial_t = \frac{\partial}{\partial t} \tag{5.13}$$

$$\partial_{ij} = \frac{\partial^2}{\partial x_i \partial x_j}. \tag{5.14}$$

Since we are assuming periodic boundary conditions, we can simplify Eq. (5.10). If we take the divergence of Eq. (5.10) we obtain:

$$\begin{aligned} \partial_i(\partial_t u_i) + \partial_i(u_j \partial_j u_i) &= -\partial_i(\partial_i p) + \nu \partial_i(\partial_{jj} u_i) \\ \partial_t(\partial_i u_i) + \partial_{ij}(u_i u_j) &= -\partial_{ii} p + \nu \partial_{jj}(\partial_i u_i) \end{aligned} \tag{5.15}$$

and using Eq. (5.11), we get

$$\partial_{ij}(u_i u_j) = -\partial_{ii} p \tag{5.16}$$

which is an instance of Poisson's equation,

$$\nabla^2 p = \sigma. \tag{5.17}$$

If the function σ has a vanishing spatial average, that is if

$$\langle \sigma \rangle = \frac{1}{N_x N_y} \int_{\boldsymbol{r}} \sigma(\boldsymbol{r}) d\boldsymbol{r} = 0, \tag{5.18}$$

holds, then we can solve Poisson's equation using functions that are periodic over the domain. It follows from Eq. (5.16) that $-\partial_{ii} p$ also vanishes, yielding

$$\partial_t u_i + u_j \partial_j u_i = \nu \partial_{jj} u_i \tag{5.19}$$

Henceforth, we will use $\langle \cdot \rangle$ to denote the spatial average $\frac{1}{N_x N_y} \int_{\boldsymbol{r}} \cdot d\boldsymbol{r}$.

Now that we have the simplified Navier-Stokes Equation Eq. (5.19), we can derive the energy balance equation. First, we state without proof certain properties involving the spatial averages of functions and the vector operations of dot products and taking the curl:

$$\begin{aligned}
\langle \partial_i f \rangle &= 0 && (5.20)\\
\langle (\partial_i f) g \rangle &= -\langle f \partial_i g \rangle && (5.21)\\
\langle (\nabla^2 f) g \rangle &= -\langle (\partial_i f)(\partial_i g) \rangle && (5.22)\\
\langle \boldsymbol{u}_1 \cdot (\nabla \wedge \boldsymbol{u}_2) \rangle &= \langle (\nabla \wedge \boldsymbol{u}_1) \cdot \boldsymbol{u}_2 \rangle && (5.23)\\
\langle \boldsymbol{u}_1 \cdot \nabla^2 \boldsymbol{u}_2 \rangle &= -\langle (\nabla \wedge \boldsymbol{u}_1) \cdot (\nabla \wedge \boldsymbol{u}_2) \rangle && (5.24)
\end{aligned}$$

With these properties, we can now present and derive the energy balance equation,

$$\frac{dE}{dt} = -2\nu\Omega \tag{5.25}$$

where

$$\begin{aligned}
E &= \frac{1}{2} \langle |\boldsymbol{u}|^2 \rangle && (5.26)\\
\Omega &= \frac{1}{2} \langle |\omega|^{\mathbf{2}} \rangle && (5.27)\\
\omega &= \nabla \wedge \boldsymbol{u}. && (5.28)
\end{aligned}$$

That is, E is the mean energy and Ω is the mean enstrophy or the spatial average of the vorticity, which in two dimensions is a scalar. In a domain with periodic boundary conditions, isolate the time-derivative term on the left-hand side of Eq. (5.19), multiply through by u_i and take the spatial average over the domain:

$$\begin{aligned}
\partial_t u_i &= -u_j \partial_j u_i + \nu \partial_{jj} u_i\\
u_i \partial_t u_i &= u_i u_j \partial_j u_i + \nu u_i \partial_{jj} u_j\\
\langle u_i \partial_t u_i \rangle &= \langle u_i u_j \partial_j u_i + \nu u_i \partial_{jj} u_j \rangle.
\end{aligned}$$

Rearranging the spatial average on the left-hand side yields

$$\partial_t \langle \frac{u_i u_i}{2} \rangle = \langle u_i u_j \partial_j u_i + \nu u_i \partial_{jj} u_j \rangle$$

and then using Eqs.(5.22) and (5.11), we can rearrange the first term on the right-hand side such that it vanishes:

$$\begin{aligned}\partial_t\langle\frac{u_iu_i}{2}\rangle &= \langle u_i(u_i\partial_j u_j) + \nu u_i\partial_{jj}u_j\rangle \\ &= \langle u_i(-(u_i\partial_j u_j)) + \nu u_i\partial_{jj}u_j\rangle \\ &= \langle u_i(-(u_i\cdot 0)) + \nu u_i\partial_{jj}u_j\rangle \\ &= \langle \nu u_i\partial_{jj}u_j\rangle.\end{aligned}$$

Now recast the above result into vector notation and apply Eq. (5.24) to obtain

$$\begin{aligned}\frac{d\langle\frac{1}{2}|\boldsymbol{u}|^2\rangle}{dt} = \frac{dE}{dt} &= \nu\boldsymbol{u}\nabla^2\boldsymbol{u} \\ &= \nu(-(\langle(\nabla\wedge\boldsymbol{u})\cdot(\nabla\wedge\boldsymbol{u})\rangle)) \\ &= \nu(-\langle\nabla\wedge\boldsymbol{u})^2\rangle \\ &= -2\frac{1}{2}\nu\langle|\omega|^{\mathbf{2}}\rangle \\ &= -2\nu\langle\frac{1}{2}|\omega|^{\mathbf{2}}\rangle \\ &= -2\nu\Omega.\end{aligned}$$

Note that $\frac{dE}{dt}$ is often denoted by ε and is called the mean energy dissipation per unit mass. We can use the energy balance equation to assess how accurately a given Lattice Boltzmann Method simulates fluid behavior.

Recall that there is a scalar factor of $\frac{1}{\tau}$ in the collision term L_j common to both the discrete-time and continuous-time Lattice-Boltzmann equations. As might be inferred, τ magnifies the nonlinear effect of the collision operator on the right-hand side of the discrete-time and continuous-time Lattice Boltzmann equations; a smaller τ magnifies the nonlinear effects due to the collision term and ultimately makes the fluid more likely to become turbulent given a sufficiently large forcing term.

We showed earlier that the factor $\frac{1}{\tau}$ and ν are related. In Succi [8], the explicit relation between ν and τ is

$$\nu = \frac{2(\tau - \frac{1}{2})}{6}$$

for the discrete-time Lattice BGK method. For the continuous-time Lattice Boltzmann Equation the relation is

$$\nu = \frac{2(\tau)}{6} = \frac{1}{3}\tau.$$

To test these relations and to begin some comparative analysis of the discrete-time and continuous-time Lattice Boltzmann equations, we applied the force field described in Eq. (5.6) for both the discrete-time and continuous-time Lattice Boltzmann equations on identical domains, using the same force magnitude. We also set the theoretical viscosity to be 0.25 for both the discrete-time and the continuous-time Lattice Boltzmann equation simulations by supplying a relaxation time of $\tau = 1.25$ for the discrete case and $\tau = 0.75$ for the continuous case. Thus the parameter set for the discrete case comprised

- relaxation time $\tau = 1.25$
- force magnitude $R_0 = 0.0005$
- grid size $N_x = N_y = 101$

and the continuous case parameter set comprised

- relaxation time $\tau = 0.75$
- force magnitude $R_0 = 0.0005$
- grid size $N_x = N_y = 101$
- Runge Kutta time step $H = 0.125$

The Incompressible Navier Stokes Equations from which we derived the energy balance equation do not include a forcing term, so in performing our validity test we must turn the force off before taking measurements. After a period of time in which we applied force to the fluid, we turned the force off by setting the magnitude R_0 to zero. Then we calculated the energy dissipation rate per unit mass $\varepsilon = \frac{dE}{dt}$, the mean enstrophy Ω and the ratio $N = \varepsilon / (-2\Omega)$ for each subsequent iteration. We plotted the graph of N against time (measured in steps or iterations) in both the discrete-time and continuous-time case.

A Well-Behaved Forcing Term

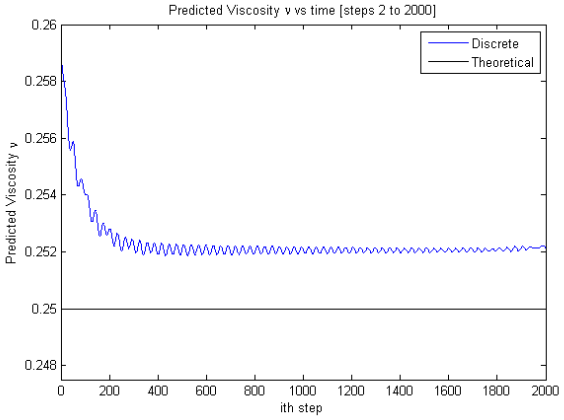
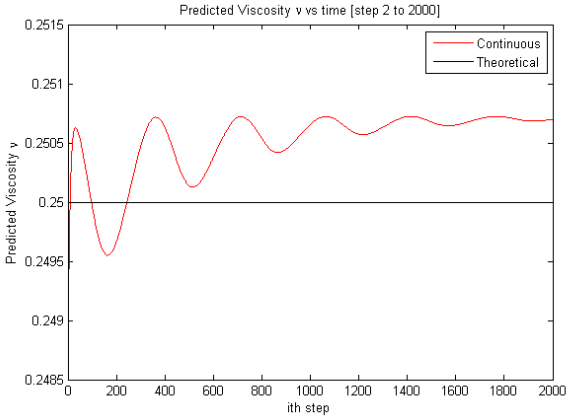


Figure 5.1: (a) Discrete case.



(b) Continuous case.

In Fig. (5.1a), the predicted viscosity does a reasonably good job of approximating the theoretical viscosity of 0.25 - the graph of the predicted viscosity decreases to 0.252 from about 0.259 within 200 discrete time steps and oscillates there for 1800 steps. The amplitude of the oscillations also seems to be decreasing with time. The discrepancy between the predicted and theoretical viscosity is 0.002 in the discrete case.

In Fig. (5.1b) the predicted viscosity for the continuous case seems to have larger-amplitude oscillations than the predicted viscosity of the discrete case. This effect is due to

scaling; the largest amplitude of the oscillations occur early within steps 0-200, where it is about 0.001.

Combining the images into one and re-scaling gives Fig. (5.2) below. Here, the discrete-case predicted viscosity graph is the top graph and the continuous-case predicted viscosity graph starts slightly below the theoretical viscosity line of 0.25. We see that the discrete-case graph features higher-frequency oscillations compared to the graph of the continuous-case, but in both cases the amplitude of the oscillations seems to be roughly the same.

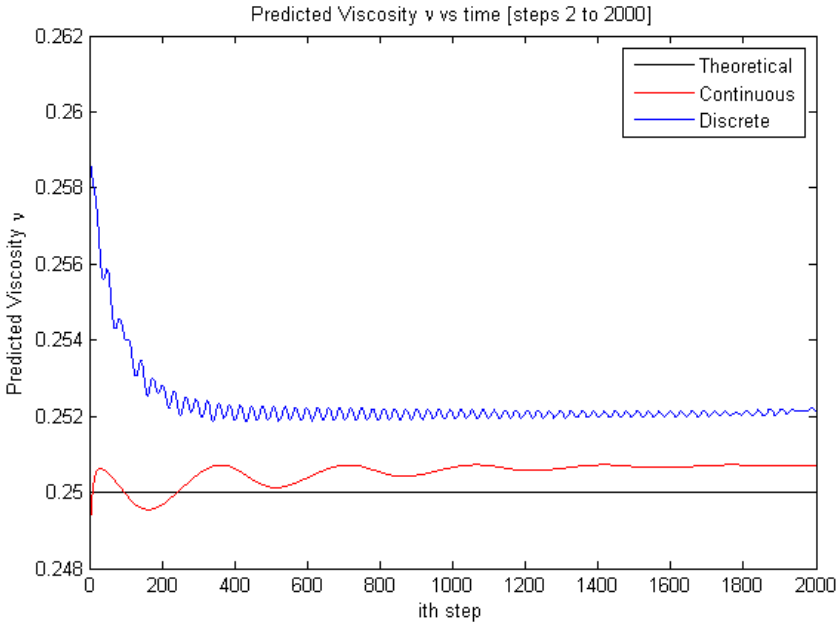


Figure 5.2: Simultaneous Plot of Discrete and Continuous cases.

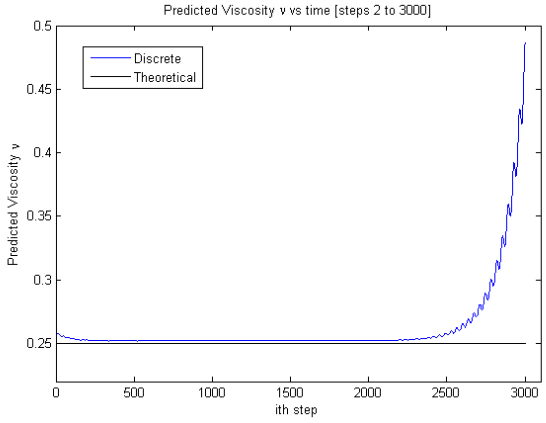
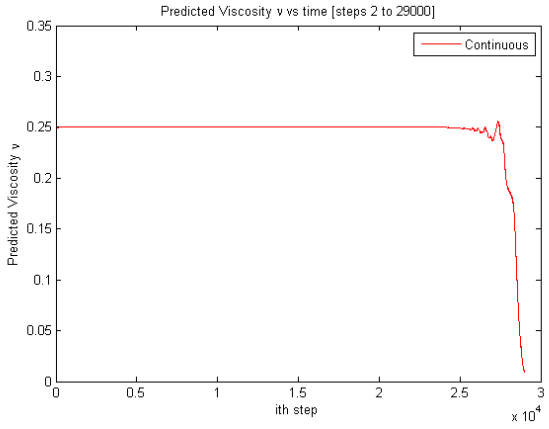


Figure 5.3: (a) Discrete case.



(b) Continuous case.

In Fig. (5.3a) above we see the plot of the discrete-case predicted viscosity for a longer period after the deactivation of the force field. We see that the predicted viscosity apparently

blows up - after hovering around 0.25 for about 2200 time steps, it rapidly increases soon afterward. In our simulation we found that after blowing up, the predicted viscosity drops quickly to zero, about 3200 steps after the deactivation of the force field.

Fig. (5.3b) shows the continuous case: for 29000 steps, the predicted viscosity is in almost perfect agreement with the theoretical viscosity. In fact, we did not include the line denoting the theoretical viscosity because the two lines coincided for 25000 steps. The scaling of the figure obscures a discrepancy from the theoretical viscosity of about 0.0005, or a quarter of the discrepancy in the discrete case. Unlike the discrete case, where the predicted viscosity blows up before ultimately vanishing, the predicted viscosity remains well-behaved for thousands of time steps before it vanishes. At worst, the discrepancy in later steps increase after about 25000 steps, but this increase is very small and short-lived; the predicted viscosity essentially settles to zero without blowing up.

Recall that in the continuous case, we defined the Runge-Kutta time step to be $H = 0.125$. Since the simulation parameters such as grid size, forcing term and effective hydrodynamic viscosity were the same for both the discrete and continuous cases, we would expect that for the same 'physical time' represented by a single step in the discrete-time simulation (i.e. from time t to time $t+1$), the continuous-time simulation would have to run for $8 = 1/0.125$ steps. In other words, if we set our basic unit of 'physical time' to be the actual time represented by a single iteration or step in the discrete-time simulation, the continuous-time simulation should take eight steps to evolve the fluid for the same unit of physical time.

After turning the force off, we see that in the discrete case we get a reasonably good predicted viscosity for about 2200 steps, and in the continuous case we get a good predicted viscosity for about 24000 steps. The ratio of 2200:24000 is about 1:11, not the 1:8 ratio we would have predicted, which suggests that the continuous-time method performs slightly better in that it is more 'stable' - that is, the computed viscosity remains well-behaved for longer 'physical time' than we would expect.

Why does the predicted viscosity in the discrete case blow up before going to zero, when the predicted viscosity in the continuous case goes to zero without blowing up? The predicted viscosity is a ratio of two quantities which both eventually go to zero after the deactivation of the force field, so it would seem that in calculating the value of the ratio the computer would eventually end up dividing by zero. We do not have a good answer for this at the moment, so we will leave it for future discussion.

At face value, the images above suggest that the continuous-time method not only yields better-behaved simulations because the continuous-time method exhibited lower-amplitude oscillations and more stable long-term behavior compared to the discrete-time method. The images above also suggest that the continuous-time method exhibits a higher degree of fidelity to theoretical predictions in that the discrepancy of the computed viscosity is closer to the theoretical viscosity.

Robustness of the method to different forcing terms

We briefly investigated whether the continuous-time method was robust to different forcing terms. The idea for this came from the fact that our first attempts at validity testing made comparisons between the discrete-time Lattice Boltzmann Equation as it was implemented using OpenLB (an open-source software package developed by Jonas Lätt for Lattice Boltz-

mann methods) and the continuous-time Lattice Boltzmann Equation as it was implemented by our program.

In the discrete case, using OpenLB and a particular set of parameters produced Fig. (5.4) below:

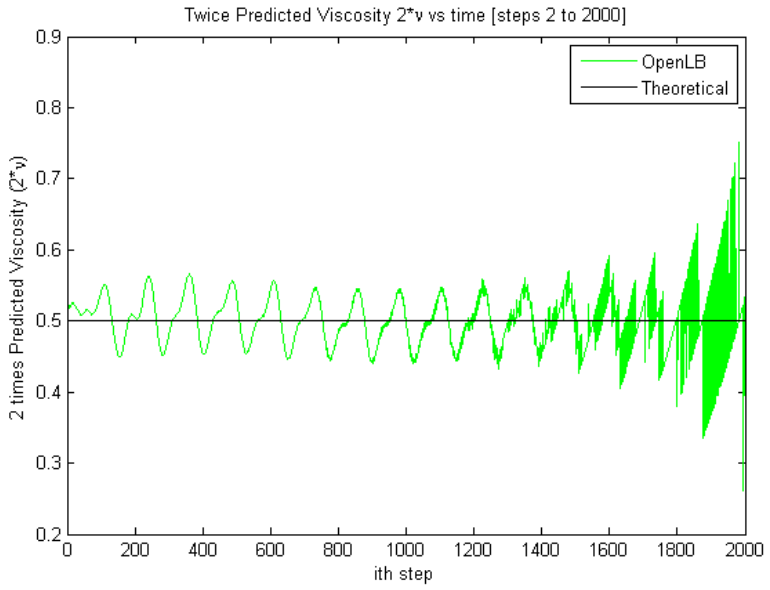


Figure 5.4: Discrete case with OpenLB

The force term used in the OpenLB simulation was defined as

$$\boldsymbol{R}(\boldsymbol{r},t) = R_0\left(\sin\left(\tilde{x}\right)\sin\left(\tilde{y}\right), \cos\left(\tilde{x}\right)\cos\left(\tilde{y}\right)\right) \tag{5.29}$$

where

$$\tilde{x} = \frac{2\pi x}{\tilde{N}} \text{ and } \tilde{y} = \frac{2\pi y}{\tilde{N}}.$$

$\tilde{N}$ represented the resolution of the lattice - the number of points in each of the x- and y-directions that represented a given characteristic length L of the physical domain. $\tilde{N}$ was chosen because the OpenLB simulation was meant to replicate the behavior of an actual physical simulation where the forcing term followed a particular spatial pattern; there is no mathematical significance to putting $\tilde{N}$ in the denominator.

We examined whether the function we wrote to implement the discrete-time Lattice Boltzmann Equation produced the same results as the OpenLB program if we used the forcing term in Eq. (5.29). Our own implementation of the discrete case featured more oscillations, and each oscillation had a minimum amplitude of about 0.3 - larger than the maximum amplitude of about 0.2 for most of the OpenLB implementation.

The parameter set for this pair of simulations was the same as that for the previous pair, with the exception of the modified forcing term and a larger force magnitude of $R_0 = 0.00812$. This parameter set corresponded to the parameter set used in the OpenLB program to generate Fig. (5.4). We present the results of our validity test for the modified force term in Fig. (5.5)-(5.7).

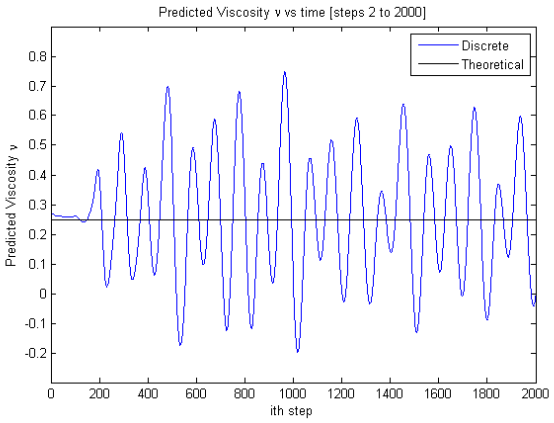
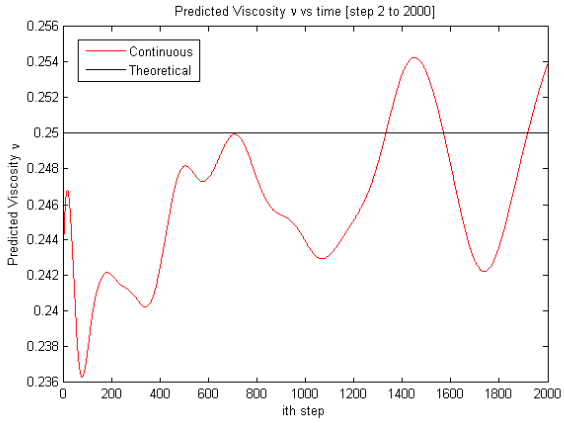


Figure 5.5: (a) Discrete case.



(b) Continuous case.

The two images for the discrete and continuous cases above present the graph of the ratio of the computed viscosity and the theoretical viscosity line of $\nu = 0.25$ for 2000 steps. We note that our implementation of the discrete case seems at first glance to be a poorer cousin of the OpenLB program - the amplitude of the oscillations in Fig. (5.5a) is on average at least 0.3, larger than the smallest amplitude in the OpenLB case of about 0.1, and more oscillations occur within 2000 steps in our implementation than in the OpenLB case.

In comparing the discrete and continuous cases in our implementation, we observe that, given the parameter set, the discrete case features oscillations of higher, more regular frequency and higher amplitude than the continuous case. The comparison is particularly stark in Fig. (5.6) below - the simultaneous plot for 8000 steps of discrete and continuous cases below shows that the graph of the computed viscosity from the continuous case is nearly flat compared to the graph of the computed viscosity from the discrete case.

The amplitude of the oscillations in the discrete case decreases slightly, but it still remains several times that of the continuous case. Mindful of the 'conversion rate' in the actual physical time elapsed per iteration between the discrete and continuous cases, we will look at the longer-term behavior of both cases later.

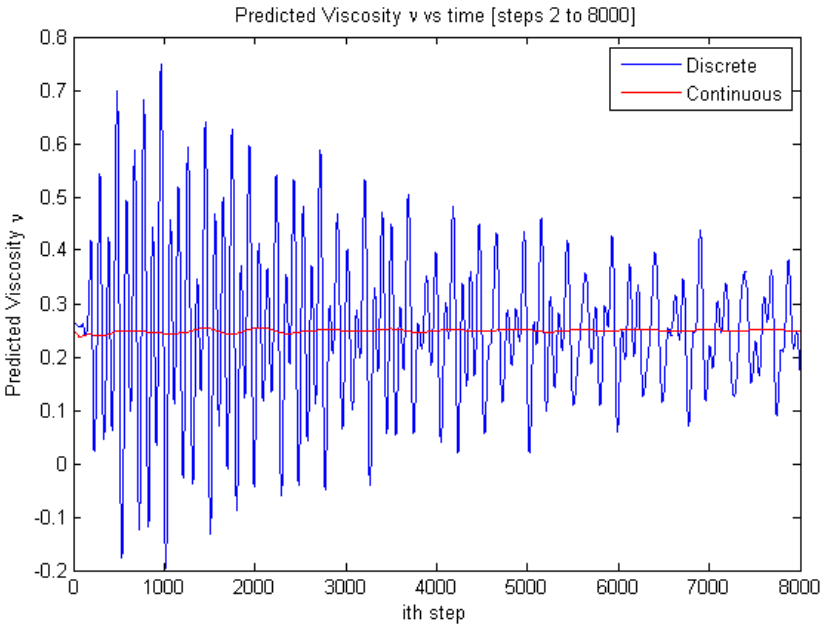
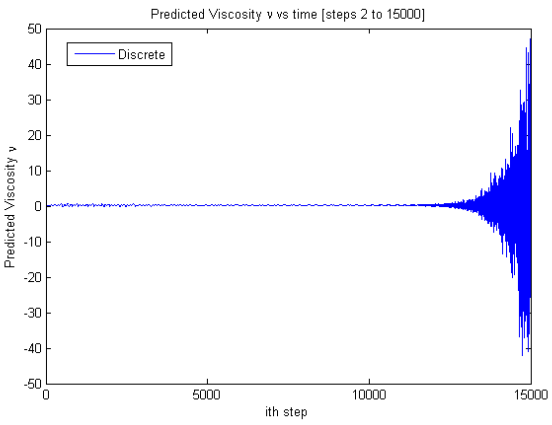


Figure 5.6: Simultaneous Plot of Discrete and Continuous cases.

In Figs. (5.7a, 5.7b) below we plot the computed viscosity for the discrete case (15000 steps) and the continuous case (30000 steps). Initial observations show that in about 12000 steps, the computed or predicted viscosity in the discrete case begins to blow up, while in 30000 steps, the predicted viscosity in the continuous case exhibits an increase in amplitude as it appears to decrease to zero. The range of the graph in the discrete case is -50 to 50, and in the continuous case it is 0.1 to 0.4.



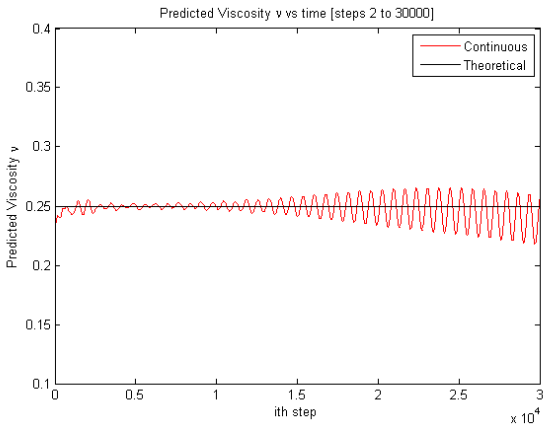


Figure 5.7: (a) Discrete case. (b) Continuous case.

Since the Runge-Kutta time step is $H = 0.125$, we again expect a 1:8 ratio of discrete-time steps to continuous-time steps for the same unit of physical time elapsed in one discrete-time step. We did not have enough readings to measure the time ratio in this case as we did before, where we found that the continuous-time case stayed steady for a longer period of time than

that implied by the 1:8 ratio. The decrease of the predicted viscosity in the continuous case below the theoretical value of $\nu = 0.25$ beginning at about 25000 steps suggests that in this case, the continuous method does not demonstrate the same long-term stability as it did before.

Our discrete-case implementation oscillates around the theoretical hydrodynamic viscosity for much longer than the OpenLB program - in Fig. (5.4) the predicted hydrodynamic viscosity starts to blow up within 2000 steps, whereas in our discrete-case implementation the oscillations remain stable and actually damp out slightly in 8000 steps as in Fig. (5.6), with catastrophic blow-up occuring at about 12000 steps in Fig. (5.7a). We are not sure why this is the case, but this might suggest that our naïve implementation of the discrete-time Lattice Boltzmann Equation is prone to more oscillatory behavior, even as it seems to enjoy an advantage in long-term stability over the OpenLB program.

Comparing Fig. (5.5a) with Fig. (5.5b) supports our earlier observation that the continuous-time Lattice Boltzmann Equation yields better-behaved simulations. There are fewer oscillations than in the discrete case, and the largest amplitude of oscillations in the continuous case within 2000 steps is 0.01 or one thirtieth of the smallest amplitude of oscillation in the discrete case.

Nevertheless, Fig. (5.7b) shows the vulnerability of the continuous-time Lattice Boltzmann Equation to less friendly parameter sets. Compared to Fig. (5.3b), the oscillations do not damp out over time, even within 8000 steps; instead, the amplitude of the oscillations grows from 0.01 to about 0.04 In Fig. (5.7b) over 30000 steps.

Our results from validity testing have shown that, on the basis of two different simulation parameter sets, the continuous-time Lattice Boltzmann Equation offers more stable long-term behavior over the discrete-time Lattice Boltzmann Equation in that oscillations do not grow quickly. A related benefit comes in the form of increased fidelity to the Incompressible Navier Stokes Equations, in that the continuous-time Lattice Boltzmann Equation provides more accurate results in the validity tests we described above. However, in its simplest implementation, the continuous-time Lattice Boltzmann Equation is not perfect; one can tweak the parameter set to generate more oscillatory behavior. We do not know why the graph of computed viscosity is oscillatory nor what affects the frequency of oscillations.

5.3 Performance

We analyze the performance of our implementation of the continuous-time Lattice Boltzmann Equation here, making comparisons with the discrete-time Lattice Boltzmann Equation when it is useful to illustrate the advantages and disadvantages to using the continuous-time Lattice Boltzmann Equation.

Storage

Our preliminary implementation of the fourth-order Runge Kutta numerical integration scheme involved creating four rate-of-change arrays denoted $k_1 - k_4$, a temporary storage array, and the main array containing the fluid particle distributions. This comes to a total of six three-dimensional arrays. For a three-dimensional array of double-type variables that

contains only information about the fluid particle distributions, a straightforward calculation shows that the storage required per array comes to $(2^3)(100^2)(9) = 720000$ bytes, or about 7.2 megabytes of memory per array, for a total of 43.2 megabytes.

By comparison, the discrete-time Lattice Boltzmann Equation at most requires only two three-dimensional arrays - one to contain the fluid state at time t and another to contain the fluid state at time $t + \Delta t$ - so that the storage requirements come to a third of the requirements for the continuous-time Lattice Boltzmann Equation.

We sought an improved implementation of the fourth-order Runge-Kutta numerical integration scheme that could reduce the number of additional three-dimensional arrays.

$$y_{n+1} = y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4, \tag{5.30}$$

where

$$\begin{aligned} k_1 &= hf(t_n, y_n) && (5.31)\\ k_2 &= hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) && (5.32)\\ k_3 &= hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) && (5.33)\\ k_4 &= hf(t_n + h, y_n + k_3) && (5.34)\\ f(t_n, y_n) &= \frac{d}{dt}y_n(t_n). && (5.35) \end{aligned}$$

In implementing the Runge-Kutta method for the continuous-time Lattice Boltzmann Equation, we consider $y_n(t_n)$ to be the fluid state or the array of particle distributions at time at time t_n and the function $f(t_n, y_n)$ the aggregate function formed from applying the continuous-time Lattice Boltzmann Equation to every particle distribution function $f_j(\boldsymbol{r}, t)$. One might consider k_1-k_4 as representing the rate-of-change arrays that are used to form a final weighted-average rate-of-change array, analogous to a $\Delta y_n(h)$, that is added to y_n to get an approximation of y_{n+1}.

The functions required to perform Runge-Kutta integration are not too difficult to program. All we need is a function that computes the rate-of-change array for a particular array containing the fluid distributions, and functions to perform the linear operations of scalar multiplication and addition on the vector space of three-dimensional arrays, by applying the same operations of scalar multiplication and addition to individual particle distributions.

In our preliminary implementation, we evaluated k_1-k_4 and stored them separately in four rate-of-change arrays. We also used a temporary storage array to contain the argument array for the function $f(y_n)$. For example, in evaluating k_2, we stored $y_n + \frac{1}{2}k_1$ in the temporary storage array and then used the entries in the temporary storage array to calculate the entries in the rate-of-change array k_2. Finally, when k_1-k_4 had been evaluated, they were individually weighted according to Eq. (5.30) and then added sequentially to the primary array, y_n.

The advantage of the preliminary method is that it is simple to understand. However, for simulations that require a large number of grid points in each linear dimension (a relevant concern especially when scaling the program to three-dimensional domains that require four-dimensional arrays), it seems reasonable to demand that the program makes the most efficient

use of computing resources, and so we will present a method for optimizing the program with respect to memory use.

To reduce the number of arrays needed for each call to the Runge-Kutta function, we decided to rearrange the order of operations. Before reordering, the function initially proceeds as follows:

- evaluate k_1 using y_n, store k_1
- evaluate k_2 using k_1 and y_n, store k_2
- evaluate k_3 using k_2 and y_n, store k_3
- evaluate k_4 using k_3 and y_n, store k_4
- add $\frac{1}{6}k_1$ to y_n
- add $\frac{1}{3}k_2$ to $(y_n + \frac{1}{6}k_1)$
- add $\frac{1}{3}k_3$ to $(y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2)$
- add $\frac{1}{6}k_4$ to $(y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3)$.

We observe that after it is evaluated, k_1 is only used in evaluating k_2 and when it is added to y_n. The intervening steps of evaluating k_3 and k_4 do not require k_1. One might be tempted to add $\frac{1}{6}k_1$ to y_n after evaluating k_2, but unless a copy of y_n is stored elsewhere in some temporary placeholder array, this would prevent the evaluation of k_3 and k_4 due to the explicit dependence of k_3 and k_4 on y_n. Fortunately, this incorrect approach is not far from an approach that works in principle and requires only three arrays in addition to y_n instead of five. To avoid confusion, we will call these arrays a, b and c. We include a segment of the modified function below to illustrate the main idea:

- store a copy of y_n in a
- use $\mathsf{a} = y_n$ to evaluate k_1, which is stored in b
- add $\frac{1}{6}\mathsf{b} = \frac{1}{6}k_1$ to y_n which now becomes $(y_n + \frac{1}{6}k_1)$
- add $\mathsf{a} = y_n$ to $\frac{1}{2}\mathsf{b} = k_1$ (so that $\mathsf{b} = \frac{1}{2}k_1 + y_n$) to evaluate k_2, which is stored in c
- add $\frac{1}{3}\mathsf{c} = \frac{1}{3}k_2$ to $(y_n + \frac{1}{6}k_1)$, which now becomes $(y_n + \frac{1}{6}k_1 + \frac{1}{3}k_2)$
- add $\mathsf{a} = y_n$ to $\frac{1}{2}\mathsf{c} = \frac{1}{2}k_2$ (so that $\mathsf{c} = y_n + \frac{1}{2}k_2$) to evaluate k_3, which is stored in b and overwrites k_1

Proceeding in this way makes the Runge-Kutta integration step a little more fragmented, but it makes better use of each array; once a rate-of-change array k_i has been used to evaluate k_{i+1} and added to y_n, it is overwritten.

Having written a new function that does the Runge-Kutta integration, we wanted to know whether the change has any effect on the ability of the program to simulate fluid behavior. From an array containing a certain initial configuration of fluid distributions, we saved two

arrays: one that had been computed from an initial-condition array using the unmodified Runge-Kutta method and the other that had been computed from the same initial-condition array using the modified Runge-Kutta method.

The mass of the absolute value of the difference between the two saved arrays vanished, showing that the distributions of the saved arrays agreed everywhere. Thus the modified Runge-Kutta method yielded exactly the same result as the unmodified method with less storage. Interestingly, the modified Runge-Kutta method appeared to enjoy a slight advantage in speed over the unmodified method, which we attribute to the fact that with the modified method all the data is concentrated in fewer locations in the computer memory compared to the unmodified method (which uses more arrays and thus has to work over a larger 'area' in computer memory).

Operations

Any complete analysis of a numerical scheme should take into account of the number of floating-point operations (flops) involved, both to understand the intensity of each component of the numerical scheme and to identify possible modifications to the method that can optimize the use of computing resources.

From Eq. (3.35), we see that calculating $\frac{df_j}{dt}$ can be decomposed into evaluating the streaming, collision and forcing terms. We will use the convention that the elementary operations of addition, subtraction, multiplication and division count for one flop each - i.e. assignment of values to a variable is not considered as taking one flop. We will not take into account any efficiencies due to symmetry and other factors first.

- The streaming term contributes two flops: one flop from evaluating $[f_j(\boldsymbol{r} - \boldsymbol{c}_j, t) + f_j(\boldsymbol{r} - \boldsymbol{c}_j, t)]$ and one flop from multiplication by $\frac{1}{2}$;
- the forcing term contributes five flops: three flops from evaluating the dot-product of two length-two vectors, one flop from multiplication by a factor of $\frac{1}{6}$ and one flop from adding the forcing term itself;
- the collision term contributes X+3 flops: one flop from evaluating $[f_j^{(eq)} - f_j]$, one flop from division by τ, X flops from evaluating $f_j^{(eq)}$ and one flop from adding the collision term to the streaming term.

Recall the jth equilibrium distribution is given by

$$f_j^{(eq)} = \rho w_j \left[1 + 3\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right) + 4.5\left(\boldsymbol{c}_j \cdot \boldsymbol{u}\right)^2 - 1.5\left(\boldsymbol{u} \cdot \boldsymbol{u}\right)\right].$$

Thus,

- subtracting the dot product $1.5(\boldsymbol{u} \cdot \boldsymbol{u})$ takes five flops - three to compute the dot product, one for multiplication by 1.5 and one for subtraction;
- adding the quantity $4.5(\boldsymbol{c}_j \cdot \boldsymbol{u})^2$ takes six flops - three to compute the dot product, one to square the dot product, one to multiply by 4.5 and another for addition;
- adding the dot product $3(\boldsymbol{c}_j \cdot \boldsymbol{u})$ takes five flops;

- multiplying by ρ and w_j takes two flops.

The total for evaluating one $f_j^{(eq)}$ is thus eighteen, so for the D2Q9 model this would amount to at most 162 flops per lattice site. Fortunately we can take advantage of the underlying symmetry of the D2Q9 lattice to reduce the flops per lattice site to forty-five flops per lattice site. The flop-count can be reduced using the observation that every vector $\boldsymbol{c}_i$ has its additive inverse $-\boldsymbol{c}_i$ in the lattice. Thus when computing dot products we merely need to note whether we should perform addition or subtraction.

The reductions are not all entirely 'free' in that not all of them arise due to symmetry - some are due to storing quantities that are common to every $f_j^{(eq)}$ and that can be reused, such as the sum $(1-1.5\boldsymbol{u}\cdot\boldsymbol{u})$. However, the additional demand on storage is small compared to the yield in flop-count reduction.

We now return to our computation of the flop count for evaluating $\frac{df_j}{dt}$. With X=45, evaluating the collision term contributes forty-eight flops for a total of fifty-five flops per lattice site. Thus computing one rate-of-change array for a N_x by N_y lattice grid takes $O(50 \cdot 9 \cdot N_x \cdot N_y)$ flops, or $O(10^6)$ flops for a 100 by 100 lattice.

A single Runge-Kutta numerical integration requires four evaluations of rate-of-change arrays in addition to a few scalar multiplications and additions of three-dimensional arrays. The additions and scalar multiplications each require the identical number of $O(9 \cdot N_x \cdot N_y)$ flops or $O(10^5)$ flops for a 100 by 100 lattice grid. Evaluating a rate-of-change array is at least ten times as expensive as scalar multiplication or addition of arrays, so the total number of flops for a single Runge-Kutta integration takes $O(10^5) + O(10^6) = O(10^6)$ flops for a 100 by 100 lattice.

We investigated the flop-count for the discrete-time Lattice Boltzmann Equation. On a single distribution, the discrete-time Lattice Boltzmann Equation requires two fewer steps than the continuous-time method. This is because the discrete case does away with any elementary operations altogether. A single discrete-time evolution for one lattice site thus requires fifty-three flops per distribution. On a 100 by 100 lattice this comes to $O(10^6)$ flops, which is what it took for the continuous-time method to calculate a rate-of-change array.

On a big-O basis it seems that there is no clear advantage to either method, so we calculated the number of flops required to evolve the fluid state from time t to time $t+1$. For the discrete case and the standard grid, the discrete method requires 4.77 million flops. The continuous-time method using the fourth-order Runge Kutta method is more involved:

- k_1 requires 4.78 million flops - 4.77 million flops to compute $f(y_n)$ and 10000 flops to compute $hf(y_n)$;
- k_2 requires 4.80 million flops - 10000 flops to multiply k_1 by 1/2, 10000 flops to add $\frac{1}{2}k_1$ to y_n, 4.77 million flops to compute $f(y_n+\frac{1}{2}k_1)$ and 10000 flops to compute $hf(y_n+\frac{1}{2}k_1)$;
- k_3 requires the same number of flops as k_2;
- k_4 requires 4.79 million flops - 10000 flops to add k_3 to y_n, 4.77 million flops to compute $f(y_n + k_3)$ and 10000 flops to compute $hf(y_n + k_3)$;
- y_{n+1} from k_1 - k_4 requires 80000 flops, since there are four scalar multiplications and four additions on the k_i and each elementary operation on the arrays requires 10000 flops.

Thus for the continuous case and the standard grid, the fourth-order Runge Kutta method requires 19.25 million flops. We expect that executing the single discrete-time method once takes one quarter of the time required to execute a single continuous-time method. On a simulation consisting of 60,000 steps with a given parameter set, the discrete-time method took 477 seconds, whereas the continuous-time method took 1990 seconds in approximate agreement with our calculations. The simulation was performed on a single 2.8Ghz Intel Xeon processor.

The most computationally expensive element of the continuous-time Lattice Boltzmann method lies in the evaluation of a rate-of-change array. The additional computations required for the continuous-time Lattice Boltzmann Equation are the price one must pay to have the flexibility of finding the fluid state at any time.

Furthermore, the continuous-time Lattice Boltzmann method allows one to change the time step used in the numerical integration method during the simulation. In discrete-time Lattice Boltzmann methods, the time step is fixed by the simulation parameters (see the Appendix on lattice unit conversions), so once these are determined by the user, changing the time step during the simulation is impossible. If one wished to shorten the time step, one would have to re-run the simulation using a finer lattice. The in-situ variability of the numerical integration time step offers the opportunity to do both error-based adaptive time stepping and event-based time stepping, but we will leave discussion of these methods to future work.

The continuous-time Lattice Boltzmann Equation also inherits the advantage from numerical integration schemes that, once one has the flexibility of determining the time step used for numerical integration, one uses the same number of flops for each numerical integration regardless of the time step. Thus we can demand greater accuracy in monitoring the fluid without having to pay an additional cost in terms of flops per numerical integration step. Unfortunately, we cannot evade the necessity of performing more numerical integration steps when the time step is small.

Parameter bounds

In our discussion of discrete-time and continuous-time Lattice Boltzmann Equations, we referred to a few parameters and parameter values. Without knowing the range of permissible values for each parameter it is difficult to know the range of simulations accessible to each method. In this section we will try to give a rough idea of what the bounds on parameters are.

In section (4) we analyzed the linear stability of global equilibria and in section (5.1) we analyzed the effect of a single RK step on the conservation of mass and momentum, given certain parameters. In this section we combine ideas from the two to analyze the numerical stability of our implementation of the continuous-time Lattice Boltzmann Equation. Our motivation for this is that a single step on RK is not very interesting with respect to stability, especially considering that for any given simulation to be useful one would have to execute several tens of thousands - if not a few hundred thousand - RK steps.

For this purpose we wrote a test program to see whether numerical instabilities arose in the long-term simulation for a given parameter set. If any numerical instability arose, we also recorded how long it took for the instability to emerge. In this program, we used Eq. (4.18) to

inform our criteria of numerical stability (or lack thereof): the program recorded a numerical instability as having occurred if the Mach number exceeded the value of 0.7. There are at least two variable quantities we could have used to define the Mach number, M:

$$M_1 = \frac{\sqrt{2E}}{c_s} \tag{5.36}$$

$$M_2 = \max_{\boldsymbol{r}} \|\boldsymbol{u}\|, \tag{5.37}$$

where E is the mean energy or half the average of the l^2 norm of the velocity vector $\boldsymbol{u}$ over the domain. c_s is a constant that depends only on the Lattice Boltzmann model being used, and for the D2Q9 model, c_s is $\frac{1}{\sqrt{3}}$. In this test program, we used M_1 because the program already incorporates a function that computes global data such as mean energy and mean enstrophy.

We considered a given set of parameters to be numerically stable if it did not become numerically unstable within ten thousand Runge-Kutta (RK) steps. We chose ten thousand steps as our cutoff point because from our experience, numerical instability would quickly set in given an instability-causing parameter set - within the first few thousand steps. We fixed a time step of $H = 0.68$ because this value was the largest time step we found for an initial exploration of parameter bounds using a 100 by 100 grid, a relaxation time $\tau = 0.25$ and force magnitude $R_0 = 0.0001$. (There is no particular significance to the value itself.)

On an initial testing of parameter bounds, we found that for $\tau = 0.3$ and $R_0 = 0.001$, a simulation ran successfully on a 101 by 101 grid for 10000 steps. Thus for $\tau \geq 0.3$ and $R_0 \leq 0.001$ we would expect the same. For values of $\tau \leq 0.15$ and $R_0 \geq 0.000125$, a simulation would not even complete the first 100 steps. This suggests that the bound on the parameter τ lies in the interval $(0.15, 0.3)$ for time step $H = 0.68$.

The easiest ways to decrease the lower bound of the parameter τ are to lower the force magnitude, to lower the time step to smaller values and to halve the grid size. This is because decreasing τ by a factor of two and lowering the force magnitude (which drives the magnitude of individual vectors in the velocity field) or increasing the grid size cancel each other out, as can be seen from the definition of the Reynolds number

$$Re = \frac{UL}{\nu}.$$

Decreasing the time step may also help decrease the lower bound on the relaxation time τ since it means more steps are needed to cover the same amount of time for a larger step, so given a sufficiently small H, one might 'convince' the fluid to be stable for a low value of τ.

From these rough estimates, we can be reasonably certain that as long as $\tau \geq 0.3$ and $R_0 \leq 0.001$, numerical instability should not set in on square grids of length $N_x = N_y \geq 100$.

Hybrid Methods

Both the discrete and continuous methods have advantages and disadvantages. The discrete method offers the advantage of speed - using a discrete method to evolve the fluid for one time step should take on average a quarter of the time it takes the continuous method to do the same task and the discrete method requires less storage. The continuous method makes

up for this additional time and storage demand through flexibility of varying the length of the time evolution step, higher fidelity to theoretical predictions and longer-term stability of behavior.

We wanted to investigate the possibility of having the best of both worlds. Instead of running a simulation purely using the discrete method or the continuous method, we wanted to see whether the fluid simulation would suffer any loss of fidelity by using a hybrid method that made use of both, switching from one to another when requested. Why would this be useful?

- A hybrid method would be useful if we wanted to make use of the *speed* of the discrete method to get from an initial condition to a point in the phase space quickly and using the continuous method would take a long time to get there. One could use adaptive time stepping to make each numerical integration step evolve the fluid over a longer period of time, but the bound on the time step we presented in the previous section would preclude the possibility of making time steps as large as we wished.

- A hybrid method would extend the advantage of the continuous-time method in adaptive time stepping. Instead of being limited to strictly positive time steps below the upper bound discussed previously, we would be able to take a 'time step' that was larger than the upper bound. This time step would be necessarily fixed, since the discrete method has a fixed time step.

We studied whether it was possible to switch from using the discrete method to the continuous method and vice versa, and our results show that it is possible to switch between methods, albeit with some side-effects. Using the parameter set

- Runge-Kutta time step $H = 0.25$;
- viscosity $\nu = 0.1$;
- implicit relaxation time (discrete case) $\tau = 0.8$;
- implicit relaxation time (continuous case) $\tau = 0.3$;
- force magnitude $R_0 = 0.001$;
- $N_x = N_y = 101$.

we computed 1001 steps each for the discrete and continuous cases (so in the figures below, changes in the graph of quantity always occur around the 1001th step). We recorded data such as mass density, global x- and y-momentum, energy, enstrophy and palinstrophy. In both cases, mass density was conserved to floating point accuracy. We plotted the x-momentum, y-momentum and mean energy against time for both methods (we do not include the plots for enstrophy and palinstrophy because they are similar to the plot for energy).

For this set of parameters, Fig. (5.8) below shows that there is a considerably rapid decrease in global x-momentum immediately after the switch from the continuous to the discrete time from slightly above zero to $-4 \cdot 10^{-13}$ from the 1000th to the 1200th step, whereas

there is no rapid change in the switch from the discrete to continuous time. Furthermore, the noticeable change in the former case is *away* from zero, whereas the slight change in the latter case is *towards* zero. This suggests that the continuous-time method is much better at conserving x-momentum than the discrete-time method.

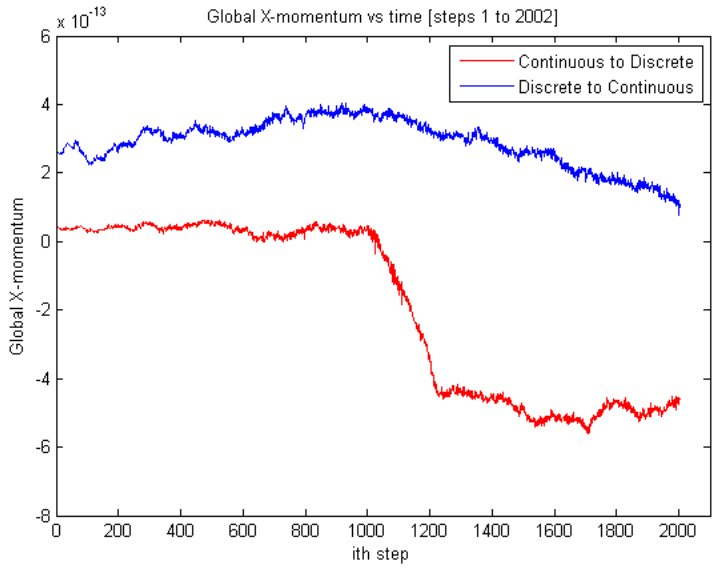


Figure 5.8: X-momentum.

In the case of global y-momentum, we also observe that the continuous-time method serves to simulate the fluid more faithfully to theoretical predictions. In Fig. (5.9) below, the plot of y-momentum for the continuous to discrete case shows that the y-momentum hovers very close to zero and then balloons upwards after the switch to discrete time. Meanwhile, y-momentum initially increases rapidly for the discrete-to-continuous case, but after the switch to continuous time, y-momentum stops its ascent at about $3.6 \cdot 10^{-12}$ and remains mostly flat at that level.

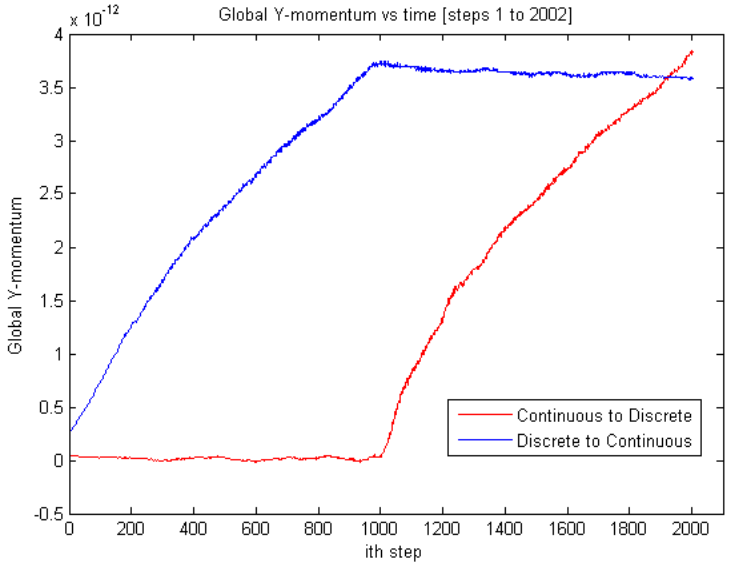


Figure 5.9: Y-momentum.

In Fig. (5.10) below, we see an interesting kink in the case for switching from the continuous to the discrete. The shallow 'well' that occurs after the switch emphasizes that in switching from the discrete-time method to the continuous-time method, one risks losing information in that some energy is lost after the switch. The energy plot for the discrete-to-continuous method does not feature any noticeable kink; instead there seems to only be a brief pause in the energy before it resumes its upward trend. We see that in both cases, for most of the discrete-time portion the energy plot is concave down and for the continuous-time portion the energy plot is concave up.

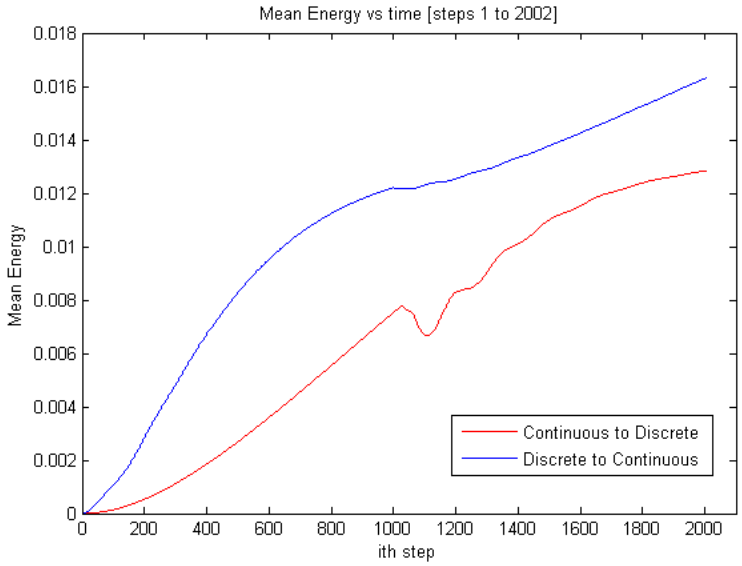


Figure 5.10: Mean Energy.

These results show that no catastrophe occurs when switching between methods. On the basis of these preliminary results, it seems that hybrid methods may be used without fear of any significant problems. However, these results suggest that switching from discrete-time to continuous-time is preferred over switching from continuous-time to discrete time, if only for the reason that the former brings the fluid to a state that is more faithful to the theoretical prediction and the latter moves the fluid away from such a state.

Finally, in order to fully exploit hybrid methods, it would be necessary to determine whether there is a conversion rate between a numerical integration of a certain time step and a single discrete-time step on the same lattice, since one would want to keep track of how many units of physical (dimensionalized) time had elapsed over the course of a simulation that used hybrid methods. It would seem that there would be a conversion rate, since the lattice determines an implicit unit of time (see the Appendix on lattice units) and every evolution of the fluid state must follow this implicit time scheme.

The results from our validity testing section suggest that an approximate conversion scheme follows a $1 : (1/H)$ ratio, where H is the Runge Kutta time step. That is, if $H = 0.125$ then it should take $8 = 1/H$ steps to evolve the fluid in the continuous-time case for the same amount of physical time as a single step in the discrete-time case. Our data was not sufficient to verify whether the conversion rate held. We leave further investigation of the conversion rate between discrete- and continuous-time methods for future work.

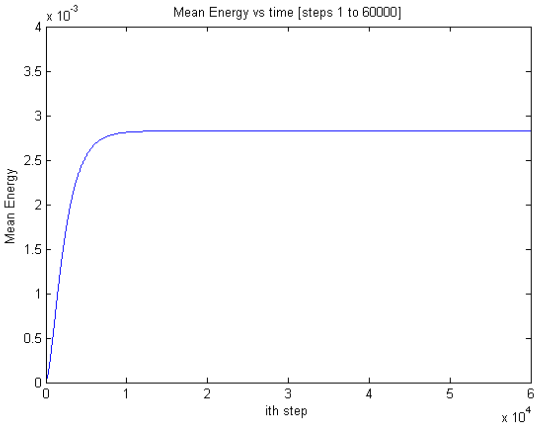
5.4 Laminar and Chaotic Regimes

The motivation for developing the continuous-time method was to be able to know the state of a fluid at any time and the intended application was finding the symbolic dynamics of turbulent fluids. So far we have applied the continuous-time method to various parameter sets and compared the continuous-time method to the discrete-time method, but we have not used the method to study laminar and chaotic flow. We attempt to remedy that problem in this subsection.

Below we present three pairs of figures, each comparing the case of laminar flow to that of chaotic flow. Both were simulated using the continuous-time Lattice Boltzmann method, with the parameter set being the same except for the relaxation time parameter τ:

- Runge Kutta time step $H = 0.125$;
- force magnitude $R_0 = 0.0005$;
- $N_x = N_y = 100$;
- relaxation time $\tau = 0.45$ (laminar case) and $\tau = 0.1$ (chaotic case)

The first pair is a plot of the mean energy against time, the second a plot of the mean enstrophy against time, and the last a plot of the discrepancy in mass from the theoretical (conserved) value. Since the energy and enstrophy are derived from the velocity field, it is not surprising that their plots against time are similar:



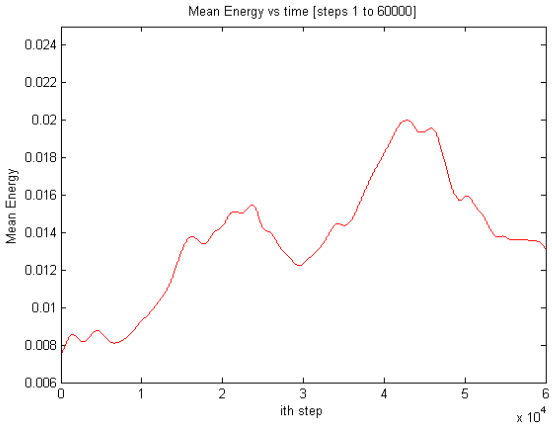
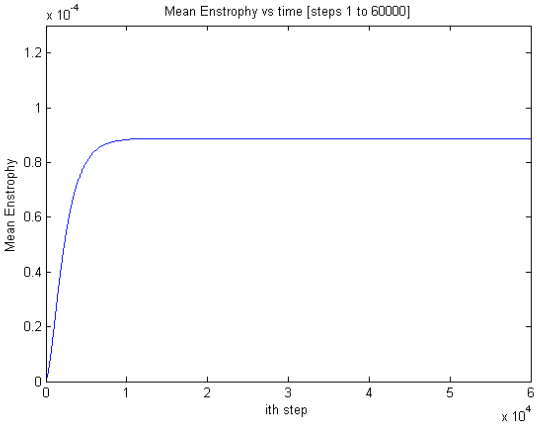


Figure 5.11: (a) Laminar Flow Energy Plot. (b) Turbulent Flow Energy Plot.



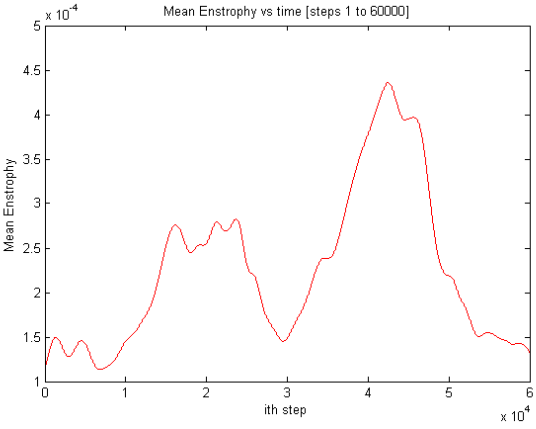
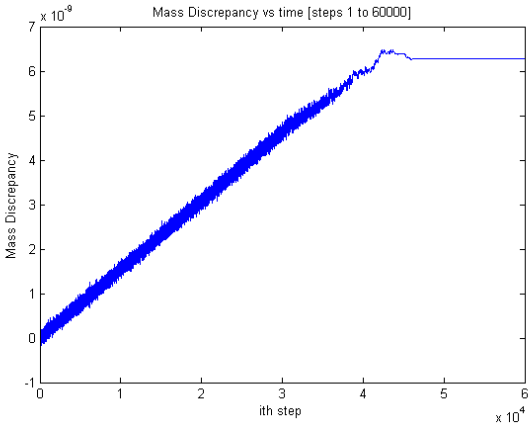


Figure 5.12: (a) Laminar Flow Enstrophy Plot. (b) Turbulent Flow Enstrophy Plot.

However, we noticed that mass discrepancy in the chaotic case was much higher than predicted. In Fig. (5.13), once the fluid has been initialized, there is an immediate mass discrepancy of $0.1 * 10^{-5}$, which exceeds numerical roundoff error - if there were numerical roundoff errors in the mass at each lattice site of $O(10^{-15})$, those errors summed over 10000 lattice sites would still only bring the total mass discrepancy to $O(10^{-11})$.



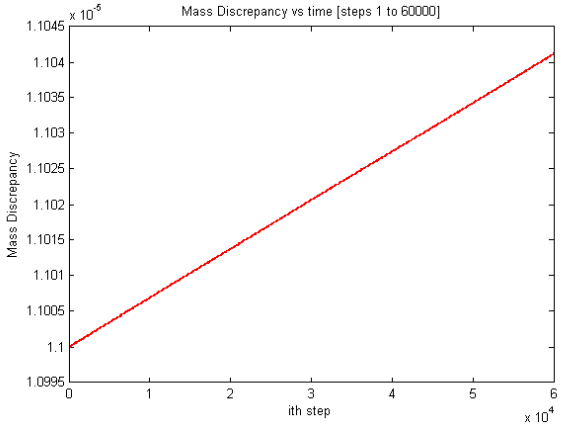


Figure 5.13: (a) Laminar Flow, Mass Discrepancy. (b) Turbulent Flow, Mass Discrepancy.

As it turns out, the extra discrepancy is mass is due to the somewhat demanding parameter set. At $\nu = 0.1/3$, the viscosity is small, so it will magnify any roundoff errors in the computation of the collision operator. The moderately high force magnitude also adds more to each f_j and the large time step H means that roundoff errors are more likely to persist in the Runge-Kutta numerical integration step.

Fortunately, Fig. (5.13) shows that, after the initial large discrepancy in mass, the discrepancy accumulates very slowly. Over the course of 60,000 steps the global mass discrepancy grows linearly to $4.5 * 10^{-8}$, or about $4.5 * 10^{-12}$ - still too large to be explained by numerical error, but of about the right size given the parameter set.

5.5 Visualizing Flows

Since we have gone to such great lengths to derive and analyze this new method for simulating fluid flows, it seemed amiss to not try and coax some fun out of our work as well. To this end we developed functions to output scalar fields (vorticity) and vector fields (velocity and palinstrophy) to .txt files, and we wrote a separate short program in Mathematica to convert these .txt files into .png files. The Mathematica program automatically puts the magnitude of the vector field as the background in each vector field image.

These visual representations of chaos serve an important function, namely that they allow for some rough comparison of fluid flows when the time-series representations alone would not suffice. One time series plot may look very similar to or very different from another, but by looking at movies of the scalar or vector field representations of the flow, one may have a better idea of whether two flows are similar.

From there the next step would be to develop a movie that displayed not only the evolution of the scalar or vector field but also the position of the fluid state along a time-series. For example, a movie consisting of vorticity scalar fields played in sequence might be accompanied by a smaller display at the bottom that indicated the 'location' of the fluid state on the time-series of plot of mean enstrophy. This would greatly aid in identifying patterns

(if they existed) or correspondences between discrete-time simulations and continuous-time simulations.

We present static image pairs for velocity, vorticity and palinstrophy - the left image corresponds to the scalar/vector field during a transient phase and the right image corresponds to the same field during a chaotic phase. Note that since we are studying fluid flow in two-dimensional domains, we defined palinstrophy as the gradient of vorticity (vorticity is a scalar for fluids in two-dimensional domains).

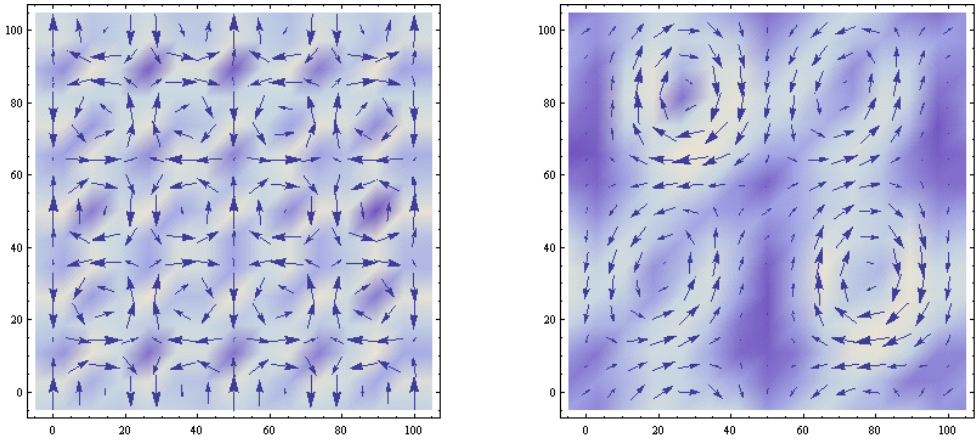


Figure 5.14: (a) Velocity, transient phase. (b) Velocity, chaotic phase.

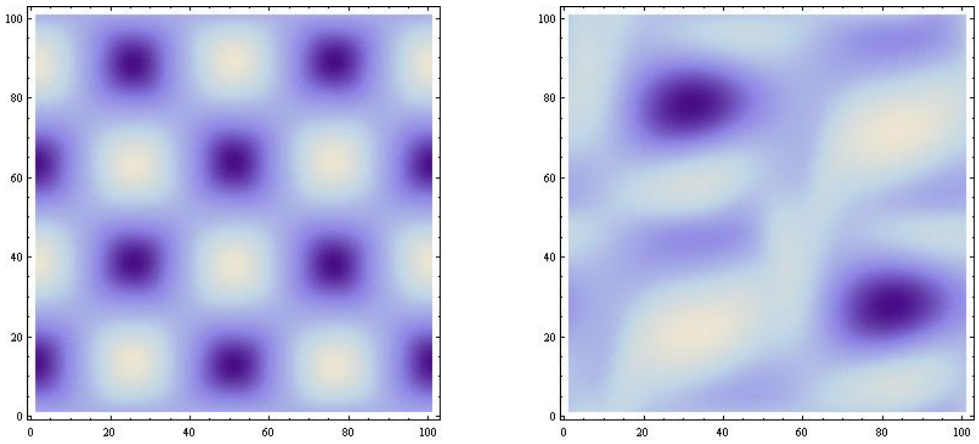


Figure 5.15: (a) Vorticity, transient phase. (b) Vorticity, chaotic phase.

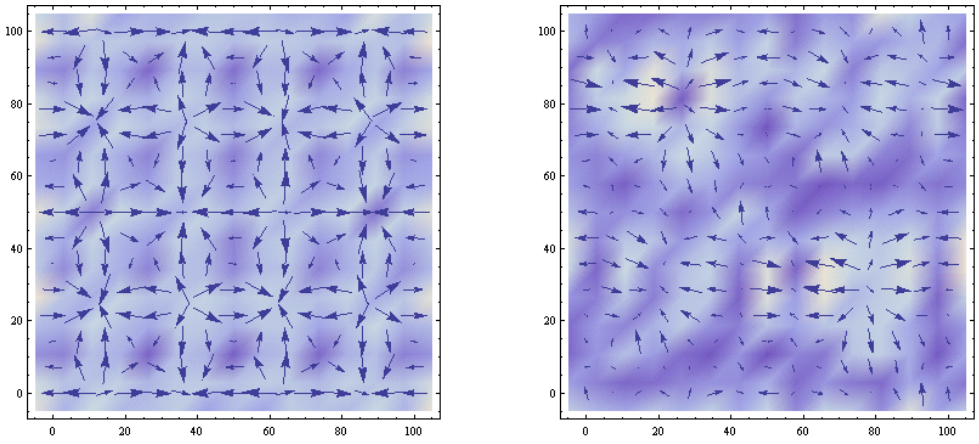


Figure 5.16: (a) Palinstrophy, transient phase. (b) Palinstrophy, chaotic phase.

5.6 Higher-accuracy Streaming

In the continuous-time Lattice Boltzmann Equation

$$\frac{df_j(\boldsymbol{r},t)}{dt} = \frac{1}{2}\left[f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t)\right] + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right]$$

the streaming operator is

$$\frac{f_j(\boldsymbol{r}-\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\boldsymbol{c}_j,t)}{2}.$$

We could rewrite this in a general form as

$$S_\alpha = \frac{f_j(\boldsymbol{r}-\alpha\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+\alpha\boldsymbol{c}_j,t)}{2\alpha} \tag{5.38}$$

with $\alpha = 1$, and here it is apparent that the streaming operator takes a form similar to that of a centered finite difference operator for the *spatial* derivative $\partial f/\partial \mathbf{x}$ at time t:

$$D_\alpha = \frac{f(\mathbf{x}+\alpha\mathbf{h},t) - f(\mathbf{x}-\alpha\mathbf{h},t)}{2\alpha\mathbf{h}}. \tag{5.39}$$

We will drop the vector notation for $\mathbf{x}$ and $\mathbf{h}$ for this discussion and write instead

$$D_\alpha = \frac{f(x+\alpha h,t) - f(x-\alpha h,t)}{2\alpha h}, \tag{5.40}$$

with the understanding that x and h are in fact vectors and that f is a real scalar function that takes in one vector-valued and one scalar-valued argument.

Returning to the general form of the streaming operator, we could modify the operator using different $\alpha \in \mathbb{N}$. For $\alpha = 2$ we have:

$$S_2 = \frac{f_j(\boldsymbol{r}-2\boldsymbol{c}_j,t) - f_j(\boldsymbol{r}+2\boldsymbol{c}_j,t)}{4}. \tag{5.41}$$

The similarity between the streaming operator and a centered finite difference is due to the fact that the DTLBE is derived from the Boltzmann Equation, which describes the behavior of gas molecules and has a spatial derivative term $\partial f/\partial \mathbf{x}$ where the f describe the distributions of particles and $\mathbf{x}$ is the position vector.

We want to modify the streaming operator so that we can obtain a formula that approximates the spatial derivative to a high degree of accuracy. Taylor expanding Eq. (5.40) in αh gives

$$D_\alpha = \frac{1}{2\alpha h}\sum_j^\infty \frac{1}{j!} f^{(j)}(x)(\alpha h)^j \left[1^j - (-1)^j\right]. \tag{5.42}$$

When j is even, the term $[1^j - (-1)^j]$ vanishes, and when j is odd the term $[1^j - (-1)^j]$ has the value 2. Thus only odd terms survive, and the equation above may be further simplified:

$$\begin{aligned} D_\alpha &= \frac{1}{\alpha h}\sum_{j=0}^\infty \frac{f^{(2j+1)}(x)}{(2j+1)!}(\alpha h)^{2j+1} \\ &= \sum_{j=0}^\infty \frac{f^{(2j+1)}(x)}{(2j+1)!}(\alpha h)^{2j} \end{aligned} \tag{5.43}$$

Our goal is to compute the spatial derivative $\partial f/\partial x$. On the lattice, the spatial derivative becomes $\partial f/\partial h$, since the particle distributions exist only on the lattice sites. Since computers perform operations in a discrete manner, we can only use finite linear combinations of discrete difference operators to approximate the spatial derivative. In other words, to approximate the spatial derivative to a high degree of accuracy, we want to find an integer A and weights W_α such that

$$\sum_{\alpha=1}^A D_\alpha W_\alpha = \frac{df}{dh} = f^{(1)}. \tag{5.44}$$

Now if we substitute the definition of D_α from Eq. (5.43) into $\sum_{\alpha=1}^A D_\alpha W_\alpha$ we obtain

$$\sum_{\alpha=1}^A D_\alpha W_\alpha = \sum_{\alpha=1}^A \left(\sum_{j=0}^\infty \frac{f^{(2j+1)}(x)}{(2j+1)!}(\alpha h)^{2j}\right) W_\alpha, \tag{5.45}$$

and we may reverse the order of summation so that

$$\sum_{\alpha=1}^A D_\alpha W_\alpha = \sum_{j=0}^\infty \left(\sum_{\alpha=1}^A \frac{(\alpha h)^{2j}}{(2j+1)!} W_\alpha\right) f^{(2j+1)}(x). \tag{5.46}$$

Since

$$f^{(1)} = \sum_{j=0}^\infty \delta_{j0} f^{(2j+1)}(x) \tag{5.47}$$

it follows that if $\sum_{\alpha=1}^A D_\alpha W_\alpha = f^{(1)}$, then

$$\sum_{\alpha=1}^A \frac{(\alpha h)^{2j}}{2j+1} W_\alpha = \delta_{j0} \tag{5.48}$$

Suppose A=1. Then by the above equation, we must have that $W_1 = 1$; thus the generalized streaming operator G_A for $A = 1$ is

$$G_1 = \sum_{\alpha=1}^{A=1} D_\alpha W_\alpha = D_1. \tag{5.49}$$

On our lattice, $D_1 = S_1$; thus $G_1 = S_1$. As our aim is to improve upon the streaming operator, we will adopt the convention that $D_\alpha = S_\alpha$ on our lattice

For A=2, find W_1 and W_2 by choosing $j = 0$,

$$\sum_{\alpha=1}^{A=2} W_\alpha = 1 \tag{5.50}$$

and $j = 1$,

$$\sum_{\alpha=1}^{A=2} \alpha^2 W_\alpha = 0. \tag{5.51}$$

We now have a linear system of $A = 2$ equations,

$$\mathcal{A}w = b,$$

where

$$\mathcal{A} = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix} \quad w = \begin{bmatrix} W_1 \\ W_2 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and the solution to this linear system is $(W_1, W_2) = (4/3, -1/3)$.

To obtain streaming operators that approximate the spatial derivative with increasing accuracy, choose larger values of $A \in \mathbb{N}$ and let $1 \leq \alpha \leq A$. This will give the matrix $\mathcal{A}$ where $(\mathcal{A}_{ij}) = j^i$ if $i \neq 1$ and $(\mathcal{A}_{ij}) = 1$ otherwise, and it also yields the vector w, whose entries $w_i = W_i$. The right-hand side vector b has entries δ_{j1} so that the first component is always 1 and the other components are always zero. This will give a system of A equations in A unknowns. Since the columns of the matrix $\mathcal{A}$ are linearly independent, there will be one vector w that solves the matrix equation.

We investigated the 'return to accuracy' from increasing A by performing a validity test for G_A, with $A = 1, 2, 3, 4$. If increasing A led to a generalized streaming operator that was more accurate with respect to the spatial derivative $\partial f/\partial h$ then we would expect that the graph of the computed or predicted viscosity would converge to the theoretical viscosity as A increased.

We ran validity tests for the weight sets. In Fig. (5.17) below we present the graphs of computed viscosity for the first four weight sets. The dashed line is the theoretical viscosity, $\nu = 0.1$, and the oscillating graph above it is the graph of the predicted viscosity for $A = 1$ - i.e. for the standard continuous-time Lattice Boltzmann Equation - and the graphs near the top are the graphs for $A = 2, 3, 4$.

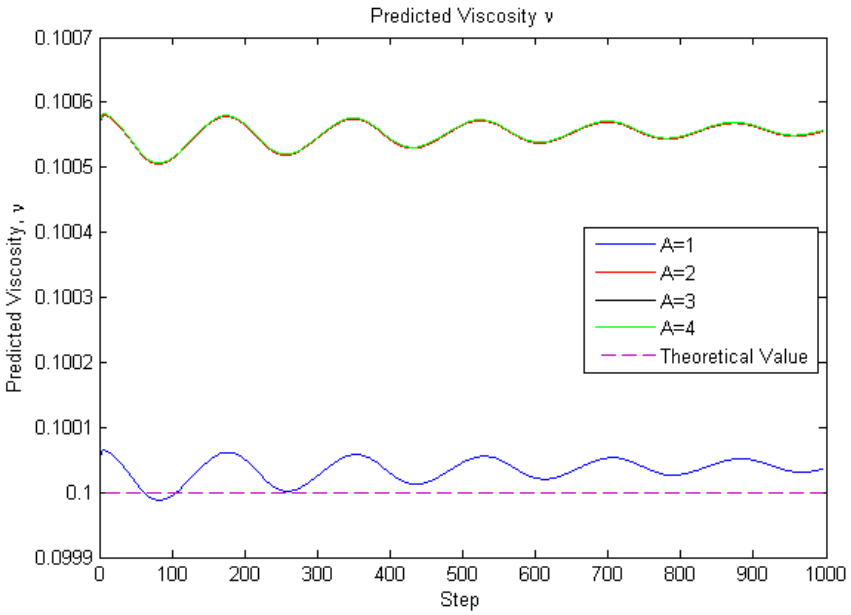


Figure 5.17: Comparison for A=1,2,3,4

We observe that the simplest generalized streaming operator (where $A = 1$) is, by this criterion, the most 'accurate' approximation to the spatial derivative. The graphs for $A = 2, 3, 4$ appear to be converging to something in Fig. (5.18) below, as the distance between successive graphs seems to decrease.

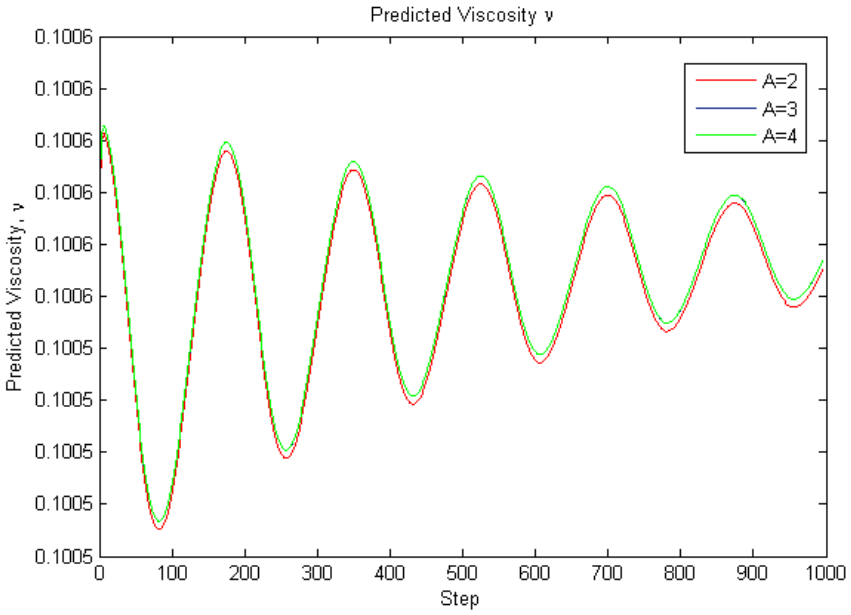


Figure 5.18: Comparison for A=2,3,4

We can measure the accuracy of the generalized streaming operator G_A with respect to

the spatial derivative by taking time averages of the predicted viscosity and measuring their distance from the theoretical value. We can also measure the 'distance' between successive graphs by taking differences of the average computed viscosity. Our results for $1 \leq A \leq 9$ are displayed below in Table 1, to five decimal places:

A	ν_A	$\lvert\nu_A - 0.1\rvert$	$\lvert\nu_A - \nu_{A-1}\rvert$
1	0.09994	5.92413 E-05	9.99408 E-02
2	0.10046	4.57171 E-04	5.16413 E-04.
3	0.10046	4.58773 E-04	1.60145 E-06
4	0.10046	4.58778 E-04	5.40753 E-09.
5	0.10046	4.58778 E-04	2.36563 E-11
6	0.10046	4.58778 E-04	3.57159 E-13.
7	0.10046	4.58778 E-04	1.65284 E-14
8	0.10046	4.58778 E-04	8.74301 E-16.
9	0.10046	4.58778 E-04	2.08167 E-16

Our preliminary results suggest that the modified streaming operator is not more accurate with respect to the Incompressible Navier Stokes Equations. There are at least three possible reasons for this:

- Programming errors (unlikely, but possible).
- The generalized streaming operator is becoming increasingly non-local: with higher values for A, the range of α increases so the generalized streaming operator is taking into account the distributions of particles that are further away from the current lattice site at which we wish to evaluate the streaming operator. That is, if we evaluate the streaming operator at $\boldsymbol{r}$, increasing the value of A means we compute S_α for higher values of α, which in turn requires computing with distributions at $\boldsymbol{r} - \alpha \boldsymbol{c}_j$. The increasing non-local nature of the generalized streaming operator conflicts with the local nature of the spatial derivative.
- Loss of information through asymptotics: Although the Boltzmann Equation does yield the Incompressible Navier-Stokes Equations under asymptotic arguments, the two equations are not the same. The graphs of the computed viscosity converge to something that is not exactly equal to the theoretical viscosity in the INSE because they may be converging to the Boltzmann Equation as A increases. This would suggest that the discrepancy between the graphs of $A = 2, 3, 4$ and the theoretical viscosity is due to the loss of information that results from applying asymptotic arguments in the derivation of the INSE from the discrete-time Lattice Boltzmann Equation.

Of these reasons, we believe that the last reason is the most plausible, as it explains why the graphs in Fig. (5.18) converge; it also explains the discrepancy between the theoretical viscosity and the graphs.

Chapter 6

Conclusion

In this paper, we derived a general continuous-time Lattice Boltzmann Equation from the discrete-time Lattice Boltzmann Equation and showed that it simulated the behavior of a fluid as well as the discrete-time Lattice Boltzmann Equation. In fact, we showed that

- the continuous-time Lattice Boltzmann Equation demonstrates long-term stable behavior and higher accuracy compared to the discrete-time Lattice Boltzmann Equation, on the basis of the validity test we devised from the energy balance equation;
- the continuous-time Lattice Boltzmann Equation is flexible in that a researcher need not recalibrate or restart her simulation when she wishes to change the time step at which the Lattice Boltzmann evolves the fluid state in time, thus overcoming a serious limitation to discrete-time Lattice Boltzmann methods;
- the continuous-time Lattice Boltzmann Equation may be implemented using any numerical integration method and it may be implemented using fixed time-stepping or adaptive (error-based or event-based) time stepping
- Higher-order spatial derivatives are easy to implement.

Weaknesses of the Lattice Boltzmann Equation are:

- using a fourth-order method to numerically integrate the continuous-time Lattice Boltzmann Equation leads to an evolution step that takes four times as long as an evolution step for the discrete-time Lattice Boltzmann method. In general, if a numerical integration method requires evaluating the function $f(x_n, y_n)$ n times, then a single evolution step using that integration method will take at least n times as long as a single evolution step with the discrete method;
- mass conservation is not guaranteed for all parameter sets - there are parameter sets for which the initial mass discrepancy is a few orders of magnitude larger than numerical roundoff error;
- 'hybrid methods' that switch between discrete-time and continuous-time methods do not lead to catastrophic instability, but the transition between the two methods is not smooth and may lead to loss of information in the fluid distributions.

For future work, we hope to find solutions to the problem of mass conservation. We also hope to study the relation between discrete-time and continuous-time methods in order to find a conversion rate between the two. In this way, we may run a discrete simulation to obtain a general idea of how a given fluid simulation evolves, and to acquire initial data in the form of rough estimates of the periodicity of a flow, for example. Subsequently we may run the continuous-time analogue with a better idea of where to 'slow' the simulation down to obtain more precise data about the flow. With these tools in hand we may begin to collect accurate data on the symbolic dynamics and ultimately on the unstable periodic orbits of turbulent fluid flow.

Bibliography

[1] Kendall E. Atkinson, *An introduction to numerical analysis*, second ed., John Wiley & Sons Inc., New York, 1989.

[2] Bruce M. Boghosian, *(Preprint) Exact Hydrodynamics of the Lattice BGK Equation*, 2010.

[3] Boghosian, Bruce M., Han C. Lie, and Sauro Succi, *(Preprint) Continuous-time Lattice Boltzmann Models*, 2010.

[4] Uriel Frisch, *Turbulence*, Cambridge University Press, Cambridge, 1995, The legacy of A. N. Kolmogorov.

[5] Harold Jeffreys, *Cartesian tensors*, Cambridge University Press, New York, 1962.

[6] Harold Jeffreys and Bertha Swirles Jeffreys, *Methods of mathematical physics*, Cambridge, at the University Press, 1956, 3d ed.

[7] Martin D. Kruskal, *Asymptotology*, (1962).

[8] Sauro Succi, *The lattice Boltzmann equation for fluid dynamics and beyond*, Numerical Mathematics and Scientific Computation, The Clarendon Press Oxford University Press, New York, 2001, Oxford Science Publications.

[9] Michael C. Sukop and Daniel T. Thorne Jr., *Lattice boltzmann modeling: An introduction for geoscientists and engineers*, first ed., Springer-Verlag, Berlin, 2006.

[10] Dieter A. Wolf-Gladrow, *Lattice-gas cellular automata and lattice Boltzmann models*, Lecture Notes in Mathematics, vol. 1725, Springer-Verlag, Berlin, 2000.

[11] Eutiquio C. Young, *Vector and tensor analysis*, second ed., Marcel Dekker, Inc., New York, 1993.

Appendix A

Tensor Analysis of D2Q9 Lattice

In this section, we will define what a tensor is and how tensors are important in Lattice Boltzmann Methods. We will restrict ourselves to analyzing the D2Q9 Lattice. Our presentation is derived largely from Young [11]and Section 3.3 of Wolf-Gladrow [10], which in turn borrows from Jeffreys [5] and Jeffreys and Jeffreys [6].

What is a tensor, and why are they useful? The set of tensors is a set of mathematical objects that encompasses familiar ones such as scalars, vectors and matrices. However, there are tensors that are neither scalars, vectors nor matrices but rather their 'higher-dimensional' analogues. Knowing scalars and vectors is a good way to begin understanding tensors, though, and we will use scalars, vectors and matrices to build some basic foundations.

As for the other question - why tensors are useful - the answer is that working with tensors is more convenient. Formally, tensors are a mathematical shorthand for objects that would be otherwise difficult, if not laborious, to explicitly write out. Tensors help to clarify patterns that might be obscured by tedious calculations.

Cartesian Tensors: Preliminaries

- Let $\boldsymbol{P}$ be a vector in $\mathbb{E}^2$.
- Let $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ be the set of canonical, orthonormal basis vectors corresponding to the rectangular cartesian co-ordinate system.
- Let $\{\boldsymbol{u}_1, \boldsymbol{u}_2\}$ be another set of orthonormal basis vectors. corresponding to a different rectangular cartesian co-ordinate system, but with the same origin as the first.
- In the first co-ordinate system, the vector $\boldsymbol{P}$ is represented by $\boldsymbol{P} = \sum_i x_i \boldsymbol{e}_i$.
- In the second system the vector $\boldsymbol{P}$ is represented by $\boldsymbol{P}^* = \sum_j y_j \boldsymbol{u}_j$.

Adopt the Einstein summation convention, whereby the repetition of an index on a single term indicates the term is to be summed over all possible values of the index:

$$x_i \boldsymbol{e}_i := \sum_i x_i \boldsymbol{e}_i.$$

In this case the index i is called a summation index. An index that is not a summation index (i.e. an index that is not repeated) is called a free index. In working with tensors, how the

index is represented is not important - when viewed separately, the object represented by

$$x_j y_k \quad 1 \leq j, k \leq 2$$

is the same as the object represented by

$$x_s y_t \quad 1 \leq s, t \leq 2.$$

Instead, what is important is the number of free and repeated indices and the terms they on which they appear. **Transformations**

The vector $\boldsymbol{P}$ is the *same object* regardless of the co-ordinate system - it represents the same set of instructions describing how to get from one point to another in two-dimensional space. Therefore, we must have that the vector represented by $x_j \boldsymbol{e}_j$ is the same as the vector represented by $y_i \boldsymbol{u}_i$, and in some sense it follows that $x_j \boldsymbol{e}_j = y_i \boldsymbol{u}_i$. Suppose we wish to express the y_i in terms of the x_j - we can do this by taking the dot product of both sides with $\boldsymbol{u}_k$:

$$x_j(\boldsymbol{e}_j \cdot \boldsymbol{u}_k) = y_i(\boldsymbol{u}_i \cdot \boldsymbol{u}_k) = y_i \delta_{ik} = y_k \tag{A.1}$$

Let $\alpha_{kj} := (\boldsymbol{e}_j \cdot \boldsymbol{u}_k)$. Then $y_k = \alpha_{kj} x_j$, and the partial derivative of y_k with respect to x_j is α_{kj}. Similarly, we can express the x_i in terms of the y_j:

$$x_k = x_i \delta_{ik} = x_i(\boldsymbol{e}_i \cdot \boldsymbol{e}_k) = y_j(\boldsymbol{u}_j \cdot \boldsymbol{e}_k) = y_j \alpha_{jk} \tag{A.2}$$

Combining these results, we have

$$y_k = \alpha_{kj} x_j = \frac{\partial y_k}{\partial x_j} x_j \tag{A.3}$$

$$x_k = \alpha_{jk} y_j = \frac{\partial x_k}{\partial y_j} y_j \tag{A.4}$$

If we substitute the expression for x_j into that for y_k, we obtain

$$y_k = \alpha_{kj} x_j = \alpha_{kj} (\alpha_{ij} y_i) = \alpha_{kj} \alpha_{ij} y_i. \tag{A.5}$$

In the rightmost term, the index j appears twice in $\alpha_{kj}\alpha_{ij}$, so the free indices are i and k. We know that the Kronecker Delta δ_{ik} also has the property that $\delta_{ik} y_i = y_k$. Thus $\alpha_{kj}\alpha_{ij} = \delta_{ik}$. From

$$x_k = \alpha_{jk} y_j = \alpha_{jk} (\alpha_{ji} x_i) = \alpha_{jk} \alpha_{ji} x_i \tag{A.6}$$

it also follows that $\alpha_{jk}\alpha_{ji} = \delta_{ik}$.

Cartesian Tensors: First Definition

We now have some background for our first definition. In $\mathbb{E}^2$, suppose that, given a rectangular Cartesian co-ordinate system, an object may be represented as a linear combination of the associated orthonormal basis vectors. The scalars that multiply the basis vectors are called the *components* of the object with respect to the co-ordinate system.

Suppose that we are given two distinct rectangular Cartesian co-ordinate systems (with the same origin) and two sets of components (x_1, x_2) and (y_1, y_2) to represent the same object in the different co-ordinate systems. The object is a rank 1 Cartesian tensor if, roughly speaking, its algebraic expression remains the same under co-ordinate transformations. For example, a vector in two-dimensional space requires two scalar components in order to be fully described, no matter what co-ordinate system is being used. More abstractly, we have the following definition:

Definition 1. *Suppose there is a set of 2 numbers that are specified with respect to a set of co-ordinate basis vectors. Suppose that the numbers are y_k in one co-ordinate system and x_j in another. The set of 2 numbers is a Cartesian tensor of rank 1 or a rank-1 Cartesian tensor in $\mathbb{E}^2$ if its co-ordinates transform according to the rule*

$$y_k = \alpha_{kj} x_j = \frac{\partial y_k}{\partial x_j} x_j. \tag{A.7}$$

Note that we could also have used

$$x_k = \alpha_{jk} y_j = \frac{\partial x_k}{\partial y_j} y_j. \tag{A.8}$$

In interpreting the definition, the important thing is to know that the defining property of tensors is invariance under co-ordinate transformations, and that the rank of a tensor is equivalent to the number of free indices on the set of quantities.

Knowing this, we can similarly define Cartesian tensors of rank 0 (rank 2) as sets of quantities with no free index (two free indices) that transform according to the rules above. We will use two examples later to show what Cartesian tensors of rank 0 and rank 2 are.

Why do we need Cartesian tensors? One of the main motivation for Cartesian tensors was from physics. Physicists knew that there were objects that retained certain properties regardless of how they were viewed. Using vectors as an example, if one wished to go from Boston to Berlin it did not matter whether one was in Medford or in Moscow - the vector 'Boston to Berlin' was the same.

Mathematicians and physicists found a way to incorporate this invariance of certain objects into their mathematical expressions. This freed physicists to work with abstract representations of objects without having to think about co-ordinate systems. Some examples of physical objects such as these are stress (the force acting on a unit of area) and the inertia of an object. We will refer to Cartesian tensors simply as 'tensors' from this point onwards.

As an example of what we mean when we say that a tensor allows physicists to work with abstract representations of objects, we will define inertia and show that it is a tensor. First, we have to prove that the dot product or scalar product of two vectors is a rank 0 tensor and that the Kronecker Delta is a rank 2 tensor.

Claim: The dot product of two vectors is a rank 0 tensor.

Proof. The dot product of two vectors $\mathbf{a} = a_j$ and $\mathbf{b} = b_j$, denoted $\mathbf{a} \cdot \mathbf{b}$, is calculated as $\sum_{j=1}^{2} a_j b_j$ or simply $a_j b_j$ in the Einstein notation. We will show that the dot product of a vector with itself is a rank 0 tensor, but the following proof can be easily extended for the case of any two arbitrary vectors. Let $y_k = \alpha_{kj} x_j$ describe a change of co-ordinate systems, from which follows the facts (proved earlier) that $\alpha_{kj}\alpha_{kp} = \delta_{jp}$ and that $\alpha_{kj}\alpha_{pj} = \delta_{kp}$. If $\mathbf{y} = y_k$ is a vector, then

$$\begin{aligned} y_k y_k &= \alpha_{kp} x_p \alpha_{kj} x_j \\ &= \alpha_{kp}\alpha_{kj}(x_j x_p) \\ &= \delta_{pj} x_j x_p. \\ &= x_p x_p \end{aligned} \tag{A.9}$$

Thus the dot product is invariant with respect to change of co-ordinates; its mathematical expression in one co-ordinate system is the 'same' as it is in the other co-ordinate system. Additionally, the dot product is also a tensor since it obeys a transformation rule, and since it has no free indices, it is a rank 0 tensor. □

Claim: The Kronecker Delta is a rank-2 tensor.

Proof. Given an orthonormal set of basis vectors $\{\mathbf{x}_1, \mathbf{x}_2\}$ in $\mathbb{E}^2$, we can express the Kronecker Delta as $\delta_{pq} = \mathbf{x}_p \cdot \mathbf{x}_q$ where $1 \leq p, q \leq 2$. Using the Einstein notation, we could rewrite this as $\delta_{pq} = x_{p,k}x_{q,k}$ where $x_{p,k}$ denotes the kth component of the pth basis vector. Then

$$\begin{aligned}(\delta_{pq})\alpha_{lp}\alpha_{hq} &= (x_{p,k}x_{q,k})\,\alpha_{lp}\alpha_{hq} \\ &= (\alpha_{lp}x_{p,k})(\alpha_{hq}x_{q,k}) \\ &= y_{l,k}y_{h,k} \end{aligned} \tag{A.10}$$

Note that $y_{l,k}$ and $y_{h.k}$ are the images under the co-ordinate transformation of $x_{p,k}$ and $x_{q,k}$ respectively. Some observations:

- $(p = q) \Rightarrow 0 = y_{l,k}y_{h,k}$
- $(p \neq q) \Rightarrow \alpha_{lp}\alpha_{hp} = \delta_{lh} = y_{l,k}y_{h,k}$

This shows that in the second co-ordinate system, where the basis is $\{\mathbf{y}_1, \mathbf{y}_2\}$, we have that $\delta_{lh} = \mathbf{y}_l \cdot \mathbf{y}_h$, $1 \leq l, h \leq 2$. This provides an example of how a tensor has the same algebraic expression under co-ordinate transformations, and shows that the Kronecker Delta is invariant with respect to change of co-ordinate systems.

In fact, we have proved something even stronger - the *components* of the Kronecker Delta are the same under any co-ordinate transformations. In the canonical Cartesian co-ordinate system we know that the Kronecker Delta has 1's on the diagonal and 0's everywhere else, and we have just shown that in another arbitrary Cartesian co-ordinate system, the Kronecker Delta has 1's on the diagonal and 0's everywhere else. This is a special property called *isotropism*. Finally, the Kronecker Delta has two free indices, so it is a rank-2 tensor. □

We now show how inertia in $\mathbb{R}^3$ is a rank-2 tensor. First, some preliminaries:

- Let a particle of mass m be located at (x_1, x_2, x_3) in $\mathbb{R}^3$, using the canonical co-ordinate system. The moment of inertia of the particle is a set of nine scalars $R_{ij} = m\left(\delta_{ij}x_k x_k - x_i x_j\right)$ for $1 \leq i, j \leq 3$.
- In another co-ordinate system defined by $x_i^* = \alpha_{ij}x_j$, the particle has the location (x_1^*, x_2^*, x_3^*), and its moment of inertia is a new set of nine scalars $R_{ij}^* = m\left(\delta_{ij}^* x_k^* x_k^* - x_i^* x_j^*\right)$
- Under the change of co-ordinates, we use the facts that the dot product and the Kronecker Delta are tensors - $x_k^* x_k^* = x_j x_j$ and $\delta_{ij}^* = \alpha_{ip}\alpha_{jq}\delta_{pq}$.

Then

$$\begin{aligned} R^*_{ij} &= m\left(\delta^*_{ij}x^*_k x^*_k - x^*_i x^*_j\right) \\ &= m\left[(\alpha_{ip}\alpha_{jq}\delta_{pq})(\alpha_{kr}x_r)(\alpha_{kr}x_r) - (\alpha_{ip}x_p)(\alpha_{jq}x_q)\right] \\ &= m\alpha_{ip}\alpha_{jq}\left[\delta_{pq}(\alpha_{kr}\alpha_{kr})x_r x_r - x_p x_q\right] \\ &= m\alpha_{ip}\alpha_{jq}\left(\delta_{pq}(\delta_{kk})x_r x_r - x_p x_q\right) \\ &= \alpha_{ip}\alpha_{jq}R_{pq}. \end{aligned} \tag{A.11}$$

Therefore the mathematical object that describes inertia is itself a tensor, since it obeys the transformation law. The tensor is clearly of rank 2, since it has two free indices.

From the past three examples we see that an object is a tensor if the mathematical expression for the object is the same in any Cartesian co-ordinate system. Using tensors distils the structure of a problem and frees one from having to consider what co-ordinate system is being used. If we have been a bit fast and loose with which condition implies which, it is because the requirement that the mathematical expression be the same under co-ordinate transformations is equivalent to requiring that the object obeys the transformation law. We now can give a formal definition for a rank-2 tensor:

Definition 2. *Suppose there is an object consisting of 2^2 components a_{ij} $(1 \leq i, j \leq 2)$. Given a change in co-ordinate system defined by $y_i = \alpha_{ij}x_j$, the object is a tensor of rank 2 if its components a_{ij} in the x_j co-ordinate system transform to the components a^*_{ij} in the y_i co-ordinate system according to the law*

$$a^*_{ij} = \alpha_{ip}\alpha_{jq}a_{pq} \tag{A.12}$$

In general, suppose an object consists of 2^n components $a_{s_1 s_2 \cdots s_n}$, $(1 \leq s_k \leq 2; 1 \leq k \leq n)$. Given a change in co-ordinate system defined by $y_i = \alpha_{ij}x_j$, the object is a tensor of rank n if its components $a_{s_1 \cdots s_n}$ in the x_j co-ordinate system transform to the components $a^*_{s_1 \cdots s_n}$ in the y_i co-ordinate system according to the law

$$a^*_{s_1 \cdots s_n} = \alpha_{s_1 p_1}\alpha_{s_2 p_2} \cdots \alpha_{s_n p_n}a_{s_1 \cdots s_n} \tag{A.13}$$

$(1 \leq s_k, p_k \leq 2; 1 \leq k \leq n)$

Now we show how one can create a rank n tensor in $\mathbb{E}^2$:

- Choose n rank-1 tensors, a_{s_k}, (2 components each).
- Form the *outer product* $A_{s_1 \cdots s_n} = a_{s_1} \cdots a_{s_n}$. ($2^n$ components).
- Using the change of co-ordinate system defined by $y_k = \alpha_{kj}x_j$, each tensor a_{s_k} in the first co-ordinate system is $a^*_{t_r} = \alpha_{t_r s_k}a_{s_k}$ in the second co-ordinate system.
- We want to show that $A^*_{t_1 \cdots t_n} = \alpha_{t_1 s_1} \cdots \alpha_{t_n s_n}A_{s_1 \cdots s_n}$. Then we will have shown that $A_{s_1 \cdots s_n}$ respects the transformation law and is a rank n tensor:

$$\begin{aligned} A^*_{t_1\cdots t_n} &= a^*_{t_1}\cdots a^*_{t_n} \\ &= (\alpha_{t_1 s_1} a_{s_1})\cdots(\alpha_{t_n s_n} a_{s_n}) \\ &= \alpha_{t_1 s_1}\cdots\alpha_{t_n s_n}\,(a_{s_1}\cdots a_{s_n}) \\ &= \alpha_{t_1 s_1}\cdots\alpha_{t_n s_n} A_{s_1\cdots s_n}. \end{aligned}$$

Thus $A_{s_1\cdots s_n}$ is a rank n tensor: it has n free indices and respects the appropriate transformation law for its rank. The procedure outlined above describes how one can form a rank-n tensor by forming the outer product of n rank-1 tensors.

The procedure not only gives a method for forming tensors, it also gives a method for determining whether a set of 2^n quantities represents a tensor of rank n. If the outer product of the set of quantities with a rank 1 Cartesian tensor is a Cartesian tensor of rank $n+1$, then the object is a Cartesian tensor of rank n.

For example, suppose we have an object with 2^2 components and 2 indices that is represented in one co-ordinate system by Γ_{ab} and by Γ^*_{ab} in another co-ordinate system. Suppose that we have a change of co-ordinates and that the outer product of Γ_{ab} with a vector x_k yields an object that we know to be Cartesian tensor of rank 3, say L_{abk}. Then

$$\begin{aligned} L^*_{abk} &= \Gamma^*_{ab} x^*_k \\ &= \alpha_{ap}\alpha_{bq}\alpha_{kr} L_{pqr} \\ &= \alpha_{ap}\alpha_{bq}\alpha_{kr}\Gamma_{pq} x_r \\ &= \alpha_{ap}\alpha_{bq}\Gamma_{pq}\,(\alpha_{kr} x_r) \\ &= \alpha_{ap}\alpha_{bq}\Gamma_{pq}\,(x^*_k) \\ &\Rightarrow \Gamma^*_{ab} = \alpha_{ap}\alpha_{bq}\Gamma_{pq} \end{aligned} \tag{A.14}$$

so Γ_{ab} is a Cartesian tensor of rank 2.

Some observations:

- Tensors of a fixed rank n are elements of a vector space. Adding two tensors of rank n and multiplying a rank n tensor by a scalar constant both yield tensors of rank n.
- Rank 2 tensors x_{ij} are *symmetric* if their components are unaltered under interchange of indices or *antisymmetric* if the sign of each components is reversed under interchange of indices. Given a rank 2 tensor, $(w_{ij} + w_{ji})$ is a symmetric tensor and $(w_{ij} - w_{ji})$ is antisymmetric. Note that the diagonal entries of antisymmetric tensors (that is, entries where the indices satisfy $i = j$) vanish.

Certain familiar objects from calculus are tensors:

- The divergence of a vector field u_i is a rank 0 tensor: $\frac{\partial u_i}{\partial x_i}$.
- The gradient of a scalar U is a rank 1 tensor: $\frac{\partial U}{\partial x_i}$.
- The curl or rotation of a vector field u_i is an antisymmetric rank 2 tensor: $(\frac{\partial u_k}{\partial x_i} - \frac{\partial u_i}{\partial x_k})$.

The lattice vectors $\boldsymbol{c}_j$ that we described in the beginning of this thesis may be used to define tensors associated with the lattice (that is, lattice tensors), by taking the outer

product of each lattice vector with itself to form $\boldsymbol{c}_j\boldsymbol{c}_j$, which are tensors of rank 2. We can form rank 3 and rank 4 tensors by taking outer products of each lattice vector with itself two and three times respectively.

All this discussion about tensors has been geared towards explaining the elephant in the room regarding the lattice the lattice vectors - what allows the lattice scheme to simulate the INSE? The lattice scheme is able to simulate the INSE because the lattice vectors and weights satisfy the following properties, required of *all* lattice schemes:

$$\sum_j^Q w_j = 1 \tag{A.15}$$

$$\sum_j^Q w_j \boldsymbol{c}_j = \mathbf{0} \tag{A.16}$$

$$\sum_j^Q w_j \boldsymbol{c}_j \boldsymbol{c}_j = c_s^2 I_2 \tag{A.17}$$

$$\sum_j^Q w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j = 0_3 \tag{A.18}$$

$$\sum_j^Q w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j = \gamma c_s^4 I_4, \tag{A.19}$$

where γ is a constant scalar (for the D2Q9 model, $\gamma = 1$), I_2 and I_4 are the isotropic rank-2 and rank-4 identity tensors respectively, and 0_3 is the third-rank zero tensor.

A tensor is *isotropic* if its components do not change regardless of how the co-ordinate axes are rotated. We've seen one isotropic rank-2 tensor already, namely the Kronecker Delta δ_{ik}. There are no nontrivial isotropic rank 1 tensors, as we will show.

- Suppose there were a nontrivial isotropic rank 1 tensor, u_i, the components of which are specified with respect to a pair of co-ordinate basis vectors.
- Rotate the axes slightly to obtain a new rank 1 tensor, $u_i^* = (\delta_{ij} - \varepsilon z_{ij})\, u_j = u_i - \varepsilon z_{ij} u_j$. [The rank 2 tensor z_{ij} is a rotation, encountered earlier, and ε is a small scalar quantity.]
- If u_i is isotropic, then we must have $u_i^* = u_i$, in which case $z_{ij}u_j = 0$.

Substituting values for i, j yields a system of equations:

$$z_{11}u_1 + z_{12}u_2 = 0$$
$$z_{21}u_1 + z_{22}u_2 = 0$$

The rotation tensor z_{ij} is antisymmetric, so its diagonal entries $z_{ii} = 0$ and its off-diagonal entries $z_{12} = (-z_{21})$ are independent. The only solution to the system of equations is then $u_1 = u_2 = 0$, which implies that the only isotropic rank 1 tensor is the rank 1 tensor with vanishing components.

Now we show that the only rank 2 isotropic tensors are all scalar multiples of the Kronecker Delta. This will help to illustrate the method for determining isotropy of tensors. Suppose u_{ik} is an isotropic rank 2 tensor. Letting z_{ij} denote the rotation tensor as before, we have:

$$\begin{aligned} u^*_{ik} &= (\delta_{ij} - \varepsilon z_{ij})(\delta_{kl} - \varepsilon z_{kl}) u_{jl} \\ &= u_{ik} - \varepsilon z_{ij}\delta_{kl}u_{jl} - \varepsilon z_{kl}\delta_{ij}u_{jl} \\ &= u_{ik} - \varepsilon z_{ij}u_{jk} - \varepsilon z_{kl}u_{il} \end{aligned} \tag{A.20}$$

where we have discarded the term premultiplied by ε^2 on account of its small magnitude. For $u^*_{ik} = u_{ik}$, we require

$$z_{ij}u_{jk} + z_{kl}u_{il} = 0. \tag{A.21}$$

Suppose $i = 1$, $k = 2$. The diagonal entries of z_{ij} vanish, so

$$z_{12}u_{22} + z_{21}u_{11} = 0 \tag{A.22}$$

but $z_{12} = (-z_{21})$ so $u_{11} = u_{22}$. If $k = i = 1$,

$$z_{12}u_{21} + z_{12}u_{12} = 0 \tag{A.23}$$

so $u_{21} + u_{12} = 0$. But if u_{ik} is isotropic, it must be symmetric, so $u_{21} = u_{12} = 0$, and $u_{ik} = 0$ if $i \neq k$ and u_{ik} is equal to a scalar if $i = k$. These rules describe the components of a Kronecker Delta tensor (up to scalar multiplication). Thus the only isotropic rank-2 tensor is a scalar multiple of the Kronecker Delta tensor.

We can show the isotropy of higher-rank tensors in a similar way, by applying a small rotation to the axes and using antisymmetry of the rotation tensor to solve for the values of the tensor. The class of isotropic tensors can be increased by allowing for both rotations as well as reflections, but we will not show that here.

With some idea of what rank n tensors are and what isotropic tensors are, we can state a theorem:

Theorem 1. *(Jeffreys and Jeffreys, 1956; Jeffreys 1965)*

1. *There are no isotropic tensors of rank 1 (vectors).*
2. *An isotropic tensor of rank 2 is proportional to $\delta_{\alpha\beta}$.*
3. *An isotropic tensor of rank 3 is proportional to $\epsilon_{\alpha\beta\gamma}$ (the Levi-Civita tensor).*
4. *There are three different (linearly independent) tensors of rank 4*

$$\delta_{\alpha\beta}\delta_{\gamma\lambda}, \quad \delta_{\alpha\gamma}\delta_{\beta\lambda}, \quad \delta_{\alpha\lambda}\delta_{\beta\gamma},$$

which can be combined to the most general form

$$T_{\alpha\beta\gamma\lambda} = a\delta_{\alpha\beta}\delta_{\gamma\lambda} + b\delta_{\alpha\gamma}\delta_{\beta\lambda} + c\delta_{\alpha\lambda}\delta_{\beta\gamma}$$

where a, b and c are arbitrary constants.

In two-dimensional Euclidean space, it can be shown that the components of an isotropic rank 4 tensor may be described by

$$\begin{aligned} T_{1111} &= T_{2222} = a + b + c, \\ T_{1122} &= T_{2211} = a \\ T_{1212} &= T_{2121} = b \\ T_{1221} &= T_{2112} = c \end{aligned} \tag{A.24}$$

where a, b, and c are scalars. We will use this fact to determine that the D2Q9 lattice satisfies the properties required of all lattice schemes.

Lattice tensors for the D2Q9 Model

Define a lattice tensor of rank n as

$$L_{\alpha_1\alpha_2\cdots\alpha_n} = \sum_i c^i_{\alpha_1} c^i_{\alpha_2} \cdots c^i_{\alpha_n} \tag{A.25}$$

where $1 \leq \alpha_j \leq 2$ for $1 \leq j \leq n$, $c^i_{\alpha_j}$ is the α_j-th cartesian component of the ith lattice displacement vector, $\boldsymbol{c}_i$. Thus for $\boldsymbol{c}_8$, $c^8_1 = 1$ and $c^8_2 = -1$. The lattice tensor $L_{\alpha_1\alpha_2\cdots\alpha_n}$ is a rank n tensor because it is the sum over i of the nth outer product of each of the lattice vectors $\boldsymbol{c}_i$. We know from the discussion above on tensors that the outer product of n rank-1 tensors or vectors is a rank-n tensor, and that tensors are closed under scalar multiplication and addition.

Recall that the lattice vectors are given by

$$\begin{aligned} &\boldsymbol{c}_0 = (0,0) \\ &\boldsymbol{c}_1 = (0,1) \qquad \boldsymbol{c}_2 = (1,0) \qquad \boldsymbol{c}_3 = (-1,0) \qquad \boldsymbol{c}_4 = (0,-1) \\ &\boldsymbol{c}_5 = (1,1) \qquad \boldsymbol{c}_6 = (-1,1) \qquad \boldsymbol{c}_7 = (-1,-1) \qquad \boldsymbol{c}_8 = (1,-1). \end{aligned}$$

The 2nd-rank D2Q9 lattice tensor is

$$L_{\alpha\beta} = \sum_j c^j_\alpha c^j_\beta$$

but since

$$\begin{aligned} \sum_j c^j_1 c^j_1 &= (c^0_1)^2 + \left[(c^1_1)^2 + (c^2_1)^2 + (c^3_1)^2 + (c^4_1)^2\right] + \left[(c^5_1)^2 + (c^6_1)^2 + (c^7_1)^2 + (c^8_1)^2\right] \\ &= 0 + (1+0+1+0) + (1+1+1+1) = 6, \end{aligned}$$

$$\begin{aligned} \sum_j c^j_2 c^j_2 &= (c^0_2)^2 + \left[(c^1_3)^2 + (c^2_2)^2 + (c^3_2)^2 + (c^4_2)^2\right] + \left[(c^5_2)^2 + (c^6_2)^2 + (c^7_2)^2 + (c^8_2)^2\right] \\ &= 0 + (0+1+0+1) + (1+1+1+1) = 6, \end{aligned}$$

and

$$\sum_j c_1^j c_2^j = c_1^0 c_2^0 + \cdots + c_1^8 c_2^8$$
$$= 0 + (0+0+0+0) + (1-1+1-1) = 0.$$

Note that

$$\sum_j c_1^j c_2^j = \sum_j c_1^j c_2^j$$

so

$$L_{\alpha\beta} = \sum_j c_{j\alpha} c_{j\beta} = 6\delta_{\alpha\beta}. \tag{A.26}$$

This shows that the second-rank lattice tensor for the D2Q9 Lattice Boltzmann Model is indeed isotropic, since the identity matrix $\delta_{\alpha\beta}$ is certainly invariant with respect to orthogonal transformations (rotations of order 2 and reflections).

We now turn to the fourth-rank lattice tensor

$$L_{\alpha\beta\gamma\lambda} = \sum_j c_\alpha^j c_\beta^j c_\gamma^j c_\lambda^j.$$

As might be inferred from the procedure above, the way to populate the fourth-rank lattice tensor is to calculate each $L_{\alpha\beta\gamma\lambda}$, where $1 \leq \alpha, \beta, \gamma, \lambda \leq 2$. We explicitly evaluate $L_{\alpha\beta\gamma\lambda}$ for a few set of values $(\alpha, \beta, \gamma\, \lambda)$ for illustrative purposes:

$$L_{1111} = \sum_j (c_1^j)^4 = (c_1^0)^4 + \cdots + (c_1^8)^4 = 0^4 + (1^4 + 0^4 + 1^4 + 0^4) + 1^4 + 1^4 + 1^4 + 1^4 = 6.$$

We see that $L_{2222} = 6$ as well.

The following calculation shows $L_{1122} = 4$:

$$L_{1122} = \sum_j (c_1^j)^2 (c_2^j)^2 = 0 + (0+0+0+0) + (1+1+1+1) = 4$$

and by the commutativity of scalar multiplication, it does not matter in which order the c_i^j are multiplied c_1^j and c_2^j appear twice, so

$$L_{1122} = L_{2211} = L_{1212} = L_{2121} = L_{1221} = L_{2112} = 4$$

What about the other components of $L_{\alpha\beta\gamma\lambda}$?

$$L_{1112} = \sum_j (c_1^j)^3 c_2^j$$
$$= 0 + (0+0+0+0) + (1-1+1-1) = 0$$

and

$$L_{1222} = \sum_j c_1^j (c_2^j)^3$$
$$= 0 + (0+0+0+0) + (1-1+1-1) = 0.$$

For all other permutations of (1,2,1,1) and (1,2,2,2) it follows then that the corresponding entry $L_{\alpha\beta\gamma\lambda}$ vanishes, by using commutativity of scalar multiplication.

Having populated the entries of this fourth-rank tensor, we must show that it is not isotropic. But this is straightforward, since we have shown that

$$\begin{aligned} L_{1111} &= L_{2222} = 6 \\ L_{1122} &= L_{2211} = 4 \\ L_{1212} &= L_{2121} = 4 \\ L_{1221} &= L_{2112} = 4 \end{aligned} \tag{A.27}$$

and $6 \neq 4 + 4 + 4$, which contradicts the requirement stated in Eq.(159). We can fix this problem by using generalized lattice tensors, whereby the outer product of the $\boldsymbol{c}_j$ are weighted. That is, a generalized lattice tensor takes the form

$$G_{\alpha_1\alpha_2\cdots\alpha_n} = \sum_j w_j c^j_{\alpha_1} c^j_{\alpha_2} \cdots c^j_{\alpha_n}. \tag{A.28}$$

Suppose we weight the vectors $\boldsymbol{c}_1$ to $\boldsymbol{c}_4$ with weight 1 and the vectors $\boldsymbol{c}_5$ to $\boldsymbol{c}_8$ with weight $\frac{1}{4}$. For the time being we leave the weight for $\boldsymbol{c}_0$ undetermined, since it does not affect any of the calculations for the components of $L_{\alpha\beta\gamma\lambda}$. Redoing our previous calculations yields

$$\begin{aligned} L_{1111} &= \sum_j w_j (c^j_1)^4 \\ &= (c^0_1)^4 + \left[(c^1_1)^4 + \cdots + (c^4_1)^4\right] + \frac{1}{4}\left[(c^5_1)^4 + \cdots + (c^8_1)^4\right] \\ &= 0^4 + [1 + 0 + 1 + 0] + \frac{1}{4}[1 + 1 + 1 + 1] = 3 \end{aligned}$$

and

$$L_{1122} = \sum_j w_j (c^j_1)^2 (c^j_2)^2 = 0 + [0 + 0 + 0 + 0] + \frac{1}{4}[1 + 1 + 1 + 1] = 1.$$

Now we have

$$\begin{aligned} L_{1111} &= L_{2222} = 3 \\ L_{1122} &= L_{2211} = 1 \\ L_{1212} &= L_{2121} = 1 \\ L_{1221} &= L_{2112} = 1 \end{aligned} \tag{A.29}$$

and so with this set of weighted lattice vectors, we get that $L_{\alpha\beta\gamma\lambda}$ is an isotropic fourth-rank tensor. Repeating our calculations for the second-rank tensor with the given weights, we can derive that $L_{\alpha\beta} = 3\delta_{\alpha\beta}$, so the second-rank lattice tensor is still isotropic.

Thus far, the only important property of the two weights used is their ratio. Any pair of weights (w_{14}, w_{58}) satisfying $w_{58} = w_{14}/4$, where w_{14} weights the lattice vectors $\boldsymbol{c}_1$-$\boldsymbol{c}_4$ and w_{58} weights the lattice vectors $\boldsymbol{c}_5$-$\boldsymbol{c}_8$ will also yield an isotropic fourth-rank tensor and

an isotropic second-rank tensor. The requirement of the D2Q9 lattice that the sum of the weights is unity forces the weights (w_0, w_{14}, w_{58}) to be $(4/9, 1/9, 1/36)$.

The other requirements that the odd-rank lattice tensors vanish follow as a result of the fact that, with the exception of the rest vector $\boldsymbol{c}_0$, the lattice vectors occur in additive-inverse pairs - so if a vector is a lattice vector, its additive inverse is also a lattice vector.

We conclude that, of the required properties of the lattice, isotropy of the second-rank and fourth-rank generalized lattice tensors determines which lattice weights are appropriate, and vanishing of the first- and third-rank generalized lattice tensors determines the lattice-vector pairs. Note that I_2 and I_4 are the rank-2 and rank-4 identity tensors in that they leave tensors of the appropriate rank unchanged (in particular, the rank-2 identity tensor in the D2Q9 case is just the 2x2 identity matrix).

Appendix B

Chapman-Enskog Analysis of Lattice Boltzmann Equation

This analysis will show how the discrete-time Lattice Boltzmann Equation (without forcing terms) can yield the Navier-Stokes Equations. It is not meant to be comprehensive, but instead aims to provide some mathematical basis for why numerically the discrete-time Lattice Boltzmann Equation can simulate fluid behavior as described by the Navier-Stokes Equations. For a proof of how the discrete-time Lattice Boltzmann Equation with forcing term yields the Navier-Stokes Equations with forcing term, see Wolf-Gladrow [10], section 5.2.5. The treatment that follows is from Boghosian [2].

The discrete-time Lattice Boltzmann Equation (formally, the Lattice Boltzmann Equation with BGK approximation) is

$$f_j(\boldsymbol{r}+\boldsymbol{c}_j, t+1) = f_j(\boldsymbol{r}, t) + \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right]. \tag{B.1}$$

We Taylor expand $f_j(\boldsymbol{r}+\boldsymbol{c}_j, t+1)$ up to first order in time and second order in $\boldsymbol{c}_j$ to get

$$f_j\left(\boldsymbol{r}+\boldsymbol{c}_j, t+1\right) = f_j, (\boldsymbol{r}, t) + 1 \cdot \frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t). \tag{B.2}$$

Rearranging the above, we get

$$f_j\left(\boldsymbol{r}+\boldsymbol{c}_j, t+1\right) - f_j, (\boldsymbol{r}, t) = 1 \cdot \frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t). \tag{B.3}$$

We can also rearrange Eq. (**??**) to yield the same left-hand side as above:

$$f_j\left(\boldsymbol{r}+\boldsymbol{c}_j, t+1\right) - f_j(\boldsymbol{r}, t) = \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right]. \tag{B.4}$$

Subtract Eq. (B.4) from Eq. (B.3) and rearranging yields

$$\frac{\partial f_j(\boldsymbol{r}, t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r}, t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r}, t) - \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r}, t) - f_j(\boldsymbol{r}, t)\right] = 0. \tag{B.5}$$

We find the mass moment of the equation by summing over j:

$$\sum_j \left\{ \frac{\partial f_j(\boldsymbol{r},t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) - \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] \right\} = 0$$

$$\frac{\partial}{\partial t}\sum_j f_j(\boldsymbol{r},t) + \sum_j \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) + \frac{1}{2}\sum_j \left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) - \frac{1}{\tau}\sum_j \left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] = 0,$$

and the last sum over j vanishes by definition of the $f_j^{(eq)}$. We can rearrange $\sum_j \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t)$:

$$\sum_j \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) = \sum_j \boldsymbol{c}_j f_j(\boldsymbol{r},t) \cdot \nabla = \boldsymbol{\pi} \cdot \nabla \tag{B.6}$$

and similarly

$$\sum_j \left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) = \nabla\nabla : \sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j(\boldsymbol{r},t) \tag{B.7}$$

so the mass moment of the equation becomes

$$\partial_t \rho + \nabla \cdot \boldsymbol{\pi} + \frac{1}{2}\nabla\nabla : \left(\sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j\right) = 0. \tag{B.8}$$

We find the momentum moment of the equation by multiplying through by $\boldsymbol{c}_j$ and summing over j,

$$\sum_j \boldsymbol{c}_j \left\{ \frac{\partial f_j(\boldsymbol{r},t)}{\partial t} + \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) + \frac{1}{2}\left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) - \frac{1}{\tau}\left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] \right\} = 0$$

$$\frac{\partial}{\partial t}\sum_j \boldsymbol{c}_j f_j(\boldsymbol{r},t) + \sum_j \boldsymbol{c}_j \boldsymbol{c}_j \cdot \nabla f_j(\boldsymbol{r},t) + \frac{1}{2}\sum_j \boldsymbol{c}_j \left(\boldsymbol{c}_j \cdot \nabla\right)^2 f_j(\boldsymbol{r},t) - \frac{1}{\tau}\sum_j \boldsymbol{c}_j \left[f_j^{(eq)}(\boldsymbol{r},t) - f_j(\boldsymbol{r},t)\right] = 0,$$

and using Eqs. (B.6,B.7) we obtain

$$\partial_t \boldsymbol{\pi} + \nabla \cdot \left(\sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j\right) + \frac{1}{2}\nabla\nabla : \left(\sum_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j f_j\right) = 0. \tag{B.9}$$

Eqs. (B.8, B.9) need to be modified so that they involve only the hydrodynamic quantities ρ and π. In order to do this we must find the second and third moments of the distribution function f_j. We will do this by an asymptotic analysis (see Kruskal [7]) whereby we decompose f_j as follows:

$$f_j = f_j^{(0)} + \epsilon f_j^{(1)} + \epsilon^2 f_j^{(2)} + O(\epsilon^3). \tag{B.10}$$

Note that $f_j^{(i)}$ does not refer to the ith derivative of f_j, but to the order-ith component of f_j.

Before going further it is instructive to explain why we use asymptotic analysis. Asymptotic analysis attempts to simplify problems by decomposing them into increasingly refined scales. Eq. (B.10) simplifies the problem of finding the value of f_j by decomposing f_j into a

order-zero approximation of f_j that contains the bulk of f_j and increasingly finer corrections $f_j^{(i)}$. The idea is that the sum $\sum_i f_j^{(i)}$ 'converges' in the limit as $i \to \infty$ to the precise value of f_j. Each of these corrections are scaled by the appropriate power of the correction parameter ϵ.

The idea of asymptotic analysis is very similar to the idea of approximating a function by its Taylor expansion, but what differentiates asymptotic analysis is that no derivatives are involved. Instead, we assume that the problem can be tackled in separate layers - first by analyzing the problem for the zeroth-order correction, then analyzing the same problem but in the first-order correction, and so on to the desired order.

In finding the second and third moments of the distribution function f_j we will focus only on the zeroth- and first-order corrections since these are all that are required to obtain the Incompressible Navier-Stokes Equations. Let ϵ be a small correction parameter. For incompressible fluids, we adopt what is known as the 'incompressible scaling limit', which assigns a power of the correction parameter to particular terms:

- terms involving density ρ scale as ϵ^2;
- terms involving momentum $\boldsymbol{\pi}$ or $\boldsymbol{c}_j$ scale as ϵ.

Additionally we will adopt 'parabolic ordering', which is implicit in the Navier-Stokes Equations and also assigns a power of ϵ to particular terms:

- terms involving a time derivative ∂_t scale as ϵ^2;
- terms involving a spatial derivative ∇ scale as ϵ.

Substitute Eq. (B.10) (ignoring the $O(\epsilon^3)$ term into Eq. (B.5) to obtain the messy equation

$$\epsilon^2 \frac{\partial}{\partial_t}\left(f_j^{(0)} + \epsilon f_j^{(1)} + \epsilon^2 f_j^{(2)}\right) + \epsilon \boldsymbol{c}_j \cdot \nabla \left(f_j^{(0)} + \epsilon f_j^{(1)} + \epsilon^2 f_j^{(2)}\right) + \frac{1}{2}\epsilon^2 (\boldsymbol{c}_j \cdot \nabla)^2 \left(f_j^{(0)} + \epsilon f_j^{(1)} + \epsilon^2 f_j^{(2)}\right) = \frac{1}{\tau}\left[f_j^{(eq)} - \left(f_j^{(0)} + \epsilon f_j^{(1)} + \epsilon^2 f_j^{(2)}\right)\right]. \tag{B.11}$$

We will only need the zeroth-order and first-order corrections $f_j^{(0)}$ and $f_j^{(1)}$. We simplify the above by keeping terms that are scaled by ϵ^i for $i \leq 2$:

$$\epsilon^2 \frac{\partial}{\partial_t} f_j^{(0)} + \epsilon \boldsymbol{c}_j \cdot \nabla f_j^{(0)} + \epsilon^2 \boldsymbol{c}_j \cdot \nabla f_j^{(1)} + \frac{1}{2}\epsilon^2 (\boldsymbol{c}_j \cdot \nabla)^2 = \frac{1}{\tau}\left[f_j^{(eq)} - f_j^{(0)}\right] - \frac{\epsilon}{\tau} f_j^{(1)} - \frac{\epsilon^2}{\tau} f_j^{(2)}. \tag{B.12}$$

Now we do a coefficient-level comparison of the terms in Eq. (B.12). There are no terms on the left-hand side that are scaled by ϵ^0, and on the right the only term is $\frac{1}{\tau}\left[f_j^{(eq)} - f_j^{(0)}\right]$, so

$$0 = \frac{1}{\tau}\left[f_j^{(eq)} - f_j^{(0)}\right] \tag{B.13}$$

and solving for $f_j^{(0)}$ yields

$$f_j^{(0)} = f_j^{(eq)} = w_j \left[\rho + \frac{1}{c_s^2} \boldsymbol{\pi} \cdot \boldsymbol{c}_j + \frac{1}{2\gamma c_s^4 \rho} \boldsymbol{\pi} \cdot \left(\boldsymbol{c}_j \boldsymbol{c}_j - c_s^2 I_2 \right) \cdot \boldsymbol{\pi} \right]. \tag{B.14}$$

The quantity γ is a quantity that varies with each Lattice Boltzmann model. For the D2Q9 model, $\gamma = 1$, but we will keep it in the calculations that follow to preserve the full generality of our analysis.

Recall that we wish to find the second and third moments of the distribution function f_j. Continuing with our asymptotic analysis, the second moment of $f_j^{(0)}$ is

$$\sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j^{(0)} = \left(\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \left[\rho + \frac{1}{c_s^2} \boldsymbol{\pi} \cdot \boldsymbol{c}_j + \frac{1}{2\gamma c_s^4 \rho} \boldsymbol{\pi} \cdot \left(\boldsymbol{c}_j \boldsymbol{c}_j - c_s^2 I_2 \right) \cdot \boldsymbol{\pi} \right] \tag{B.15}$$

but we know that odd moments (i.e. sums involving an odd number of $\boldsymbol{c}_j$) vanish, so

$$\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \left(\frac{1}{c_s^2} \boldsymbol{c}_j \cdot \boldsymbol{\pi} \right) = \frac{1}{c_s^2} \sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \cdot \boldsymbol{\pi} = 0 \tag{B.16}$$

and therefore

$$\sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j^{(0)} = \left(\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \rho + \frac{1}{2\gamma c_s^4 \rho} \left[\boldsymbol{\pi} \cdot \left(\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \cdot \boldsymbol{\pi} - c_s^2 \boldsymbol{\pi} \cdot \boldsymbol{\pi} \left(\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \right]. \tag{B.17}$$

Since the D2Q9 model possesses the tensor properties required of any valid Lattice Boltzmann model, we get

$$\begin{aligned}
\sum_j \boldsymbol{c}_j \boldsymbol{c}_j f_j^{(0)} &= c_s^2 I_2 \rho + \frac{1}{2\gamma c_s^4 \rho} \left[\boldsymbol{\pi} \cdot \left(\gamma c_s^4 I_4 \right) \cdot \boldsymbol{\pi} - c_s^2 |\boldsymbol{\pi}|^2 \left(c_s^2 I_2 \right) \right] \\
&= c_s^2 \rho I_2 + \frac{1}{2\gamma c_s^4} \left[\gamma c_s^4 \boldsymbol{\pi} \cdot I_4 \cdot \boldsymbol{\pi} - c_s^4 |\boldsymbol{\pi}|^2 I_2 \right] \\
&= \left(c_s^2 \rho - \frac{|\boldsymbol{\pi}|^2}{2\gamma \rho} \right) I_2 + \frac{1}{2\rho} \left(|\boldsymbol{\pi}|^2 I_2 + 2 \boldsymbol{\pi} \boldsymbol{\pi} \right) \\
&= \left[c_s^2 \rho + \left(1 - \frac{1}{\gamma} \frac{|\boldsymbol{\pi}|^2}{2\rho} \right) \right] I_2 + \frac{1}{\rho} \boldsymbol{\pi} \boldsymbol{\pi}.
\end{aligned} \tag{B.18}$$

The third moment can be calculated similarly:

$$\begin{aligned}
\sum_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j f_j^{(0)} &= \left(\sum_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \left[\rho + \frac{1}{c_s^2} \boldsymbol{\pi} \cdot \boldsymbol{c}_j + \frac{1}{2\gamma c_s^4 \rho} \boldsymbol{\pi} \cdot \left(\boldsymbol{c}_j \boldsymbol{c}_j - c_s^2 I_2 \right) \cdot \boldsymbol{\pi} \right] \\
&= \left(\sum_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \left[\frac{1}{c_s^2} \boldsymbol{\pi} \cdot \boldsymbol{c}_j \right] \\
&= \frac{1}{c_s^2} \boldsymbol{\pi} \cdot \left(\sum_j w_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \boldsymbol{c}_j \right) \\
&= \gamma c_s^2 \boldsymbol{\pi} \cdot I_4,
\end{aligned} \tag{B.19}$$

where we have again used the fact that odd moments vanish.

Now we need to find $f_j^{(1)}$. Matching the terms scaled by ϵ^1 on both sides of Eq. (B.12), we get

$$\boldsymbol{c}_j \cdot \nabla f_j^{(0)} = \left(\frac{1}{\tau}\right)\left[-f_j^{(1)}\right]. \tag{B.20}$$

Rearranging gives

$$f_j^{(1)} = -\tau \boldsymbol{c}_j \cdot \left[(\nabla\rho)\,\frac{\partial f_j^{(0)}}{\partial \rho} + (\nabla\boldsymbol{\pi}) \cdot \frac{\partial f_j(0)}{\partial \boldsymbol{\pi}}\right], \tag{B.21}$$

since $f_j^{(0)} = f_j(eq)$ by Eq. (B.14). The equilibrium distribution is a function of ρ and $\boldsymbol{\pi}$ and so the 'total derivative' $\nabla f_j^{(0)}$ is equal to the contributions from the partial derivatives, $(\nabla\rho)\,\frac{\partial}{\partial\rho}$ and $(\nabla\boldsymbol{\pi}) \cdot \frac{\partial}{\partial\boldsymbol{\pi}}$.

Applying the incompressible scaling limit, the term $(\nabla\rho)$ in the square brackets scales as ϵ^3. In keeping with our earlier decision to ignore terms that scale higher than ϵ^2, we may ignore it as it is too 'fine' a correction for our asymptotic analysis:

$$f_j^{(1)} = -\tau \boldsymbol{c}_j \cdot \left[(\nabla\boldsymbol{\pi}) \cdot \frac{\partial f_j(0)}{\partial \boldsymbol{\pi}}\right]. \tag{B.22}$$

Substituting in Eq. (B.14) gives

$$f_j^{(1)} = -\tau w_j \boldsymbol{c}_j \cdot (\nabla\boldsymbol{\pi}) \cdot \left[\frac{1}{c_s^2}\boldsymbol{c}_j + \frac{1}{\gamma c_s^4 \rho}\left(\boldsymbol{c}_j\boldsymbol{c}_j - c_s^2 I_2\right) \cdot \boldsymbol{\pi}\right]. \tag{B.23}$$

Again, by properties of odd moments, all but the first term inside the square brackets vanish, so the above simplifies to

$$f_j^{(1)} = -\frac{\tau}{c_s^2} w_j \boldsymbol{c}_j \cdot (\nabla\boldsymbol{\pi}) \cdot \boldsymbol{c}_j \tag{B.24}$$

and we can multiply this by $\boldsymbol{c}_j\boldsymbol{c}_j$ and sum over j to obtain

$$\begin{aligned}
\sum_j \boldsymbol{c}_j\boldsymbol{c}_j f_j^{(1)} &= \sum_j \boldsymbol{c}_j\boldsymbol{c}_j \left[-\frac{\tau}{c_s^2} w_j \boldsymbol{c}_j \cdot (\nabla\boldsymbol{\pi}) \cdot \boldsymbol{c}_j\right] \\
&= -\frac{\tau}{c_s^2} \sum_j w_j \boldsymbol{c}_j\boldsymbol{c}_j\boldsymbol{c}_j \cdot (\nabla\boldsymbol{\pi}) \cdot \boldsymbol{c}_j \\
&= -\frac{\tau}{c_s^2} (\nabla\boldsymbol{\pi}) \,:\, \left(\sum_j w_j \boldsymbol{c}_j\boldsymbol{c}_j\boldsymbol{c}_j\boldsymbol{c}_j\right) \\
&= -\tau\gamma c_s^2 (\nabla\boldsymbol{\pi}) \,:\, I_4 \\
&= -\tau\gamma c_s^2 \left[I_2 \nabla\cdot\boldsymbol{\pi} + \nabla\boldsymbol{\pi} + (\nabla\boldsymbol{\pi})^\top\right],
\end{aligned} \tag{B.25}$$

where the last equality follows by

$$\begin{aligned}
(\nabla\boldsymbol{\pi}) \,:\, I_4 &= (\partial_k \boldsymbol{\pi}_\ell)(\delta_{ij}\delta_{k\ell} + \delta_{ik}\delta_{j\ell} + \delta_{i\ell}\delta_{jk}) \\
&= \delta_{ij}\partial_k\boldsymbol{\pi}_k + \partial_i\boldsymbol{\pi}_j + \partial_j\boldsymbol{\pi}_i \\
&= (\nabla\cdot\boldsymbol{\pi})I + \nabla\boldsymbol{\pi} + (\nabla\boldsymbol{\pi})^\top.
\end{aligned}$$

Thus we have the second moment of $f_j^{(1)}$. From Eq. (B.24) we see that the third moment of $f_j^{(1)}$ will vanish because $\boldsymbol{c}_j$ appears twice in the expression for $f_j^{(1)}$, so multiplying by $\boldsymbol{c}_j\boldsymbol{c}_j\boldsymbol{c}_j$ and summing over the j will give an odd moment.

All this groundwork has laid the foundations for a brief derivation of the Navier-Stokes Equations from Eqs. (B.8, B.9). Using the approximation in Eq. (B.10) and substituting in Eq. (B.18,B.25), the equation for mass is

$$\partial_t\rho + \frac{1}{2}\nabla\nabla : \left\{\left[c_s^2\rho + \left(1-\frac{1}{\gamma}\right)\frac{|\boldsymbol{\pi}|^2}{2\rho}\right]I_2 + \frac{1}{\rho}\boldsymbol{\pi}\boldsymbol{\pi} - \tau\gamma c_s^2\left[I_2\nabla\cdot\boldsymbol{\pi} + \nabla\boldsymbol{\pi} + (\nabla\boldsymbol{\pi})^\top\right]\right\} = 0. \tag{B.26}$$

Using incompressible scaling, we multiply the above equation by the appropriate scaling terms:

$$\epsilon^4\partial_t\rho + \epsilon^2\frac{1}{2}\nabla\nabla : \left\{\epsilon^2\left[c_s^2\rho + (1-\frac{1}{\gamma})\frac{|\boldsymbol{\pi}|^2}{2\rho}\right]I_2 + \epsilon^2\frac{1}{\rho}\boldsymbol{\pi}\boldsymbol{\pi} - \epsilon^2\tau\gamma c_s^2\left[I_2\nabla\cdot\boldsymbol{\pi} + \nabla\boldsymbol{\pi} + (\nabla\boldsymbol{\pi})^\top\right]\right\}. \tag{B.27}$$

Note that we did not scale ρ by ϵ^2 if ρ appeared in the denominator of a fraction. The divergence term $\nabla\cdot\boldsymbol{\pi}$ is two orders higher than the other terms, so the divergence term dominates and we get

$$\nabla\cdot\boldsymbol{\pi} = 0. \tag{B.28}$$

In the same way, we have the equation for conservation of momentum

$$\begin{aligned}\partial_t\boldsymbol{\pi} + \nabla\cdot\left\{\left[c_s^2\rho + \left(1-\frac{1}{\gamma}\right)\frac{|\boldsymbol{\pi}|^2}{2\rho}\right]I_2 + \frac{1}{\rho}\boldsymbol{\pi}\boldsymbol{\pi} - \tau\gamma c_s^2\left[I_2\nabla\cdot\boldsymbol{\pi} + \nabla\boldsymbol{\pi} + (\nabla\boldsymbol{\pi})^\top\right]\right\} \\ + \frac{1}{2}\nabla\nabla : \left(\gamma c_s^2\boldsymbol{\pi}\cdot I_4\right) = 0;\end{aligned} \tag{B.29}$$

including the scaling terms gives

$$\begin{aligned}\epsilon^3\partial_t\boldsymbol{\pi} + \epsilon\nabla\cdot\left\{\left[\epsilon c_s^2\rho + \left(1-\frac{1}{\gamma}\right)\epsilon^2\frac{|\boldsymbol{\pi}|^2}{2\rho}\right]I_2 + \epsilon^2\frac{1}{\rho}\boldsymbol{\pi}\boldsymbol{\pi} - \tau\gamma c_s^2\left[\epsilon^2 I_2\nabla\cdot\boldsymbol{\pi} + \epsilon^2\nabla\boldsymbol{\pi} + \epsilon^2(\nabla\boldsymbol{\pi})^\top\right]\right\} \\ + \epsilon^3\frac{1}{2}\nabla\nabla : \left(\gamma c_s^2\boldsymbol{\pi}\cdot I_4\right) = 0.\end{aligned} \tag{B.30}$$

This time, we must keep all terms that scale by ϵ^i for $0 \le i \le 3$. We need to recover the Incompressible Navier-Stokes Equations, which describe the rate of change of momentum of the fluid; if we ignored higher-order terms (order three and higher), the expression $\partial_t\boldsymbol{\pi}$ would be eliminated and we would not be able to achieve our goal. However, we will ignore terms that scale higher than order three. Working with Eq. (B.29), we obtain

$$\nabla\cdot\left[c_s^2\rho + \left(1-\frac{1}{\gamma}\right)\frac{|\boldsymbol{\pi}|^2}{2\rho}\right]I_2 = \nabla P \tag{B.31}$$

where

$$P = c_s^2\rho + \left(1-\frac{1}{\gamma}\right)\frac{|\boldsymbol{\pi}|^2}{2\rho}; \tag{B.32}$$

We obtain the advection term by noticing that

$$\nabla \cdot \left(\frac{1}{\rho} \boldsymbol{\pi}\boldsymbol{\pi} \right) = \frac{1}{\rho} \nabla \cdot (\boldsymbol{\pi}\boldsymbol{\pi}) - \frac{1}{\rho^2} (\nabla \rho) \cdot \boldsymbol{\pi}\boldsymbol{\pi}, \tag{B.33}$$

and since the term $\frac{1}{\rho}(\nabla\rho) \cdot \boldsymbol{\pi}\boldsymbol{\pi}$ scales as ϵ^5 we may ignore it to recover

$$\begin{aligned} \nabla \cdot (\boldsymbol{\pi}\boldsymbol{\pi}) &= \frac{1}{\rho} \nabla \cdot (\boldsymbol{\pi}\boldsymbol{\pi}) \\ &= \frac{1}{\rho} \left[(\nabla \cdot \boldsymbol{\pi})\boldsymbol{\pi} + \boldsymbol{\pi} \cdot \nabla \boldsymbol{\pi} \right] \\ &= \frac{1}{\rho} \left[0 + \boldsymbol{\pi} \cdot \nabla \boldsymbol{\pi} \right]. \end{aligned}$$

Finally, we obtain the term with the viscosity ν:

$$\nabla \cdot \left[-\tau \gamma c_s^2 (\nabla \boldsymbol{\pi}) : I_4 \right] + \frac{1}{2} \nabla\nabla : (\gamma c_s^2 \boldsymbol{\pi} \cdot I_4) = -\nu \nabla^2 \boldsymbol{\pi} \tag{B.34}$$

where we have defined

$$\nu = \gamma c_s^2 \left(\tau - \frac{1}{2} \right). \tag{B.35}$$

Reconstituting the conservation of momentum equation using all these terms yields

$$\partial_t \boldsymbol{\pi} + \frac{1}{\rho} \boldsymbol{\pi} \cdot \nabla \boldsymbol{\pi} = -\nabla P + \nu \nabla^2 \boldsymbol{\pi}, \tag{B.36}$$

and together with Eq. (B.28) we have the Navier-Stokes Equations. As pointed out before, the constant γ is a Lattice Boltzmann model-specific number. For the D2Q9 model, $\gamma = 1$ so $P = c_s^2 \rho$. Dividing Eqs. (B.28,B.36) through by ρ yields

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla \frac{P}{\rho} + \nu \nabla^2 \boldsymbol{u} \tag{B.37}$$

$$\nabla \cdot \boldsymbol{u} = 0 \tag{B.38}$$

where the modified pressure term $\frac{P}{\rho}$ is sometimes denoted by p. These are the incompressible Navier-Stokes Eqns. (5.8,5.9) that we saw in Subsection 5.2.

Appendix C

Conversion from Lattice Units to Physical Units

The Lattice Boltzmann method involves dimensionless quantities. While dimensionless quantities are much easier to work with, it would be useful to be able to translate between the dimensionless and dimensionalized quantities. Thus we will explain how to obtain dimensionalized units (centimeters, grams and seconds) from Lattice Boltzmann quantities.

Length: When using Lattice Boltzmann simulations, the intuitive and common thing to do is to let the space unit be represented by the lattice spacing Δx. Using a square nonperiodic (periodic) grid of N points in both x- and y-directions to represent a square domain with side length L centimeters, the space unit becomes

$$S_l \equiv \Delta x = \frac{L}{N} \text{ (in centimeters).} \tag{C.1}$$

Time: The time unit for Lattice Boltzmann simulations is the time step intrinsic to the lattice - that is, it is the time required for a distribution of particles to move along one of the displacement vectors $\boldsymbol{c}_1$-$\boldsymbol{c}_8$ from one grid point to another. If C_s is the physical value of sound speed (which, according to Google search, is 340.29 meters per second or 34029 centimeters per second at sea level), and $c_s = \frac{1}{\sqrt{3}}$ is the speed of sound on the lattice then the time unit is

$$S_t \equiv \Delta t = \frac{c_s}{C_s}\Delta x \text{ (in seconds).} \tag{C.2}$$

Mass: Lattice particles are assumed to have unit mass, so the mass unit is

$$S_m \equiv m \text{ (in grams)} \tag{C.3}$$

where m is the dimensionalized mass of the fluid molecules. To find how many molecules are represented in a single LBE particle, assume that the density d satisfies $0 < d \leq 1$; since each lattice site has Q displacement vectors, the lattice 'number density' or number of molecules per lattice site is Qd. Denote by n the corresponding physical number density, or number of molecules per square centimeter (or cubic centimeter for a three-dimensional domain). Then the number of physical molecules represented by a single LBE particle on a D-dimensional domain is found by calculating

$$S_N = \frac{nS_l^D}{Qd} \text{ molecules.} \tag{C.4}$$