Lattice Boltzmann Method for Simulating Turbulent Flows

by

Yusuke Koda

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The lattice Boltzmann method (LBM) is a relatively new method for fluid flow simulations, and is recently gaining popularity due to its simple algorithm and parallel scalability. Although the method has been successfully applied to a wide range of flow physics, its capabilities in simulating turbulent flow is still under-validated. Hence, in this project, a 3D LBM program was developed to investigate the validity of the LBM for turbulent flow simulations through large eddy simulations (LES).

In achieving this goal, the 3D LBM code was first applied to compute the laminar flow over two tandem cylinders. After validating against literature data, the program was used to study the aerodynamic effects of the early 3D flow structures by comparing between 2D and 3D simulations. It was found that the span-wise instabilities have a profound impact on the lift and drag forces, as well as on the vortex shedding frequency.

The LBM code was then modified to allow for a massively parallel execution using graphics processing units (GPU). The GPU enabled program was used to study a benchmark test case involving the flow over a square cylinder in a square channel, to validate its accuracy, as well as measure its performance gains compared to a typical serial implementation. The flow results showed good agreement with literature, and speedups of over 150 times were observed when two GPUs were used in parallel.

Turbulent flow simulations were then conducted using LES with the Smagorinsky subgrid model. The methodology was first validated by computing the fully developed turbulent channel flow, and comparing the results against direct numerical simulation results. The results were in good agreement despite the relatively coarse grid. The code was then used to simulate the turbulent flow over a square cylinder confined in a channel. In order to emulate a realistic inflow at the channel inlet, an auxiliary simulation consisting of a fully developed turbulent channel flow was run in conjunction, and its velocity profile was used to enforce the inlet boundary condition for the cylinder flow simulation. Comparison of the results with experimental and numerical results revealed that the presence of the turbulent flow structures at the inlet can significantly influence the resulting flow field around the cylinder.

Acknowledgments

First and foremost, I would like to thank my adviser, Professor Fue-Sang Lien, for his encouragements and invaluable support. Completing this project would not have been possible without his guidance. I would also like to thank Natural Sciences and Engineering Research Council (NSERC) and Bombardier Aerospace for their financial support through the NSERC Industrial Postgraduate Scholarship (IPS) program. Finally, I would also like to acknowledge the Shared Hierarchical Academic Research Computing Network (SHARCNET), for the computations involved in this work were made possible by their facilities.

Dedication

To my parents, Yoshihiro and Yuko, for their unwavering support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In Computational Fluid Dynamics (CFD), one of the most challenging tasks is to accurately and efficiently simulate turbulent flow. The main aspect about turbulent flows that make them difficult to predict is that it contains a wide range of length and time scales. Although it is possible to resolve all scales of motion by employing an extremely fine mesh through a direct numerical simulation (DNS), the resulting computational costs can become astronomical for practical flows. Therefore, it is more common to use turbulence models to simplify the computation. The most widely used and practical method of such is the Reynolds Averaged Navier-Stokes (RANS) approach, where the unsteadiness of the flow is modeled. This approach has proven to work well for certain flows, but the fact that a wide range of physics is accounted for by predetermined models make them non-ideal for a universal representation of turbulent flows. Large eddy simulations (LES) on the other hand, attempts to resolve the large scale motions, and use subgrid scale models to incorporate the effects of the small scale motions. This approach is more realistic than the RANS approach, since the features of the small eddies in a flow are relatively universal, and independent of the flow geometry.

The lattice Boltzmann method (LBM) is a numerical technique derived from the Boltzmann equation and kinetic theory, and is being recognized as an alternative to the methods based on the Navier-Stokes equations for flow computations. Contrary to the conventional CFD methods that solve the Navier-Stokes equations, the LBM employs discretized particle velocity distribution functions based on microscopic fluid physics to emulate the hydrodynamic flow field. In the LBM, fictitious particles are assumed at each node on

the computational domain, where the particles are allowed to advect along restricted directions, and collide with other particles. The movement and collision rules are designed such that "coarse graining" the particle distributions will recover the weakly compressible Navier-Stokes equations. Recently, the LBM has been gaining popularity, especially with the advent of the multiple-relaxation time (MRT) model that significantly improved its numerical stability. The method has successfully been applied to flows with complex geometries, multi-phase flows, micro-fluidics, and turbulent flows [1, 11].

The primary advantages of the LBM compared to traditional CFD methods such as the finite volume method, are:

- Ease of coding.
- Suitability for parallel execution.
- Ability to incorporate complex geometries.

These features stem from the fact that LBM is derived from particle based interactions that follow simple physics with limited range, which also obviates the need to solve the elliptic Poisson equation, common to most traditional CFD methods. Its disadvantages on the other hand, include:

- Numerical instabilities when the grid resolution is low.
- Existence of compressibility effects when simulating incompressible flow.

In the context of LBM applied to turbulent flows, there have been some notable work on some fundamental flows using LES [4, 29, 52, 61, 67, 73], following the pioneering work by Hou et al. [25], who successfully incorporated the Smagorinsky subgrid model [60] to the LBM framework. However, investigations involving the applicability of LBM for LES are still at an early stage [55], and further evaluation of its accuracy and efficiency is necessary.

1.2 Objective

This thesis aims to investigate the validity of the LBM for simulating the incompressible flow over bluff bodies in both the laminar and turbulent regimes, and its implications in the context of high performance computing (HPC). In particular, the following three topics were studied:

- Analyzing the effects of 3D flow structures in the laminar flow over two tandem cylinders.

- Exploring the computational efficiency of the LBM when implemented on the graphics processing unit (GPU) for massively parallel execution.
- Evaluating the accuracy of the LBM for performing a LES on the turbulent flow over a square cylinder confined in a channel.

1.3 Thesis Structure

Background information on the LBM, including its origins and basic implementations are given in Chapter 2. Chapter 3 presents a study on the aerodynamic effects of the 3D flow structures that develop in the flow over two tandem cylinders. The content of the chapter is largely based on the author's publication that can be found in [28]. The LBM code that was developed in this thesis project was then modified to allow for a massively parallel execution on the GPU. The background information related to GPU computing, as well as the specifics of the current implementation, are outlined in Chapter 4. The GPU enabled LBM code was then applied to simulate the turbulent flow over a square cylinder confined in a channel in Chapter 5. Finally, in Chapter 6, the concluding remarks, as well as recommended future work are presented.

Chapter 2

Background

2.1 Lattice Gas Automaton

The LBM originates from the Lattice Gas Automaton (LGA) model proposed by Frisch et al. [19], which involved simple, parallelizable algorithms, capable of simulating incompressible flows. In this model, the lattice sites carry Boolean information on whether a particle exists or not. The particles each have momentum, that allows them to travel along the lattice structure. When two or more particles reach a single lattice site, it is assumed that the particles interact in a ballistic fashion, effectively altering their momenta. The LGA evolution equation for each individual lattice site can be written as:

$$n_i(\vec{r} + \vec{c_i}, t + 1) = n_i(\vec{r}, t) + C_i \tag{2.1}$$

where $n_i(\vec{r}, t)$ is a Boolean number of particles with velocity $\vec{c_i}$, and i are directions from site $\vec{r}$ to its neighboring sites. C_i is the collision operator, where $C_i \in \{-1, 0, +1\}$ for any LGA model [74]. The macroscopic variables related to the Navier-Stokes equations can be obtained by local linear operations at $n_i(\vec{r}, t)$. Equation 2.1 can be solved by starting from an initial configuration, and updating the particle locations based on two alternating steps [11]:

1. Streaming, where every particle moves along the lattice to a neighboring site in the direction of its velocity.
2. Collision, where the particles arriving at each lattice site interact and scatter according to predefined rules.

2.2 Lattice Boltzmann Equation

Although LGA successfully modeled various fluid behavior, the computed hydrodynamic flow field contained statistical noise inherent from the Boolean processes [15]. In order to solve this problem, the Boolean particle occupation variables, n_i, were replaced by its ensembled average, $f_i = \langle n_i \rangle$, which became the primitive variables constituting the Lattice Boltzmann Equation (LBE) [34]:

$$f_i(\vec{r} + \vec{c_i}, t + 1) = f_i(\vec{r}, t) + \Omega_i \tag{2.2}$$

where $f_i(\vec{r}, t)$ is the probability distribution function of the particle populations at site $\vec{r}$, and Ω_i is the collision operator. It is important to note that this equation is formulated in terms of lattice units, where both the lattice spacing and the time interval between iterations are unitary. The macroscopic variables, density ρ and velocity $\vec{u}$, are defined from the local particle distributions by their moments as:

$$\rho = \sum_{i=0}^{q-1} f_i \tag{2.3}$$

$$\vec{u} = \frac{1}{\rho} \sum_{i=0}^{q-1} \vec{c_i} f_i \tag{2.4}$$

where q is the number of lattice neighbors associated with the given node.

A common lattice structure used for the 2D LBM is referred to as the D2Q9 lattice, where the "D2" denotes the number of dimensions, and the "Q9" denotes the number of discrete velocities associated with each lattice node. For the D2Q9 lattice, the nine discrete velocities of a node are directed to its nearest neighbors (including itself), as shown in Figure 2.1. For 3D LBM, the D3Q19 lattice (Figure 2.2) is commonly used.

2.3 Single-relaxation-time Lattice Boltzmann Method

The most popular collision operator is known as the Bhatnagar-Gross-Krook (BGK) collision operator, which makes use of a single relaxation time towards local equilibrium, and is often described as the single-relaxation-time (SRT) LBM model. The collision operator for this model can be shown to be [11]:

$$\Omega_i^{BGK} = -\frac{1}{\tau}(f_i - f_i^{eq}). \tag{2.5}$$

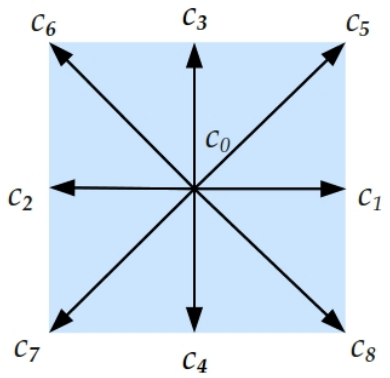

Figure 2.1: The 9 velocity components associated with each node for the D2Q9 lattice. Note that c_0 is a $(0, 0)$ vector.

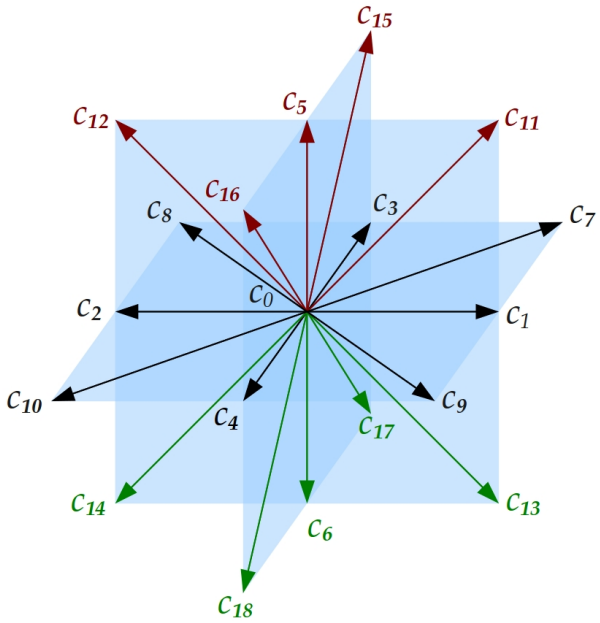

Figure 2.2: The 19 velocity components associated with each node for the D3Q19 lattice. Note that c_0 is a $(0, 0, 0)$ vector.

τ is the relaxation time, and f_i^{eq} is the local equilibrium distribution. τ defines the viscosity of the fluid, as:

$$\nu = \frac{c\delta x}{3}\left(\tau - \frac{1}{2}\right). \tag{2.6}$$

c is the lattice speed, which is defined as $\delta x/\delta t$, where δx is the lattice spacing, and δt is the time increment, and is unity for uniform lattice structures. f^{eq} is defined as:

$$f_i^{eq} = \rho w_i \left(1 + \frac{1}{c_s^2}\vec{c_i}\cdot\vec{u} + \frac{1}{2c_s^2}\boldsymbol{Q}_i : \vec{u}\vec{u}\right). \tag{2.7}$$

w_i is a constant, c_s is the speed of sound of the model, which is normally taken to be $1/\sqrt{3}$, and the tensor $\boldsymbol{Q}$ is defined as:

$$\boldsymbol{Q}_i = \vec{c_i}\vec{c_i} - c_s^2\boldsymbol{I} \tag{2.8}$$

where $\boldsymbol{I}$ is the identity matrix. The constants $q, \vec{c_i}, c_s$, and w_i are defined by the structure of the lattice. The lattice weights, w_i, are used to take into account the different magnitudes of the lattice vectors. For the D2Q9 model, the lattice weights can be shown to be:

$$\begin{aligned} w_0 &= \frac{4}{9} \\ w_{1-4} &= \frac{1}{9} \\ w_{5-8} &= \frac{1}{36}, \end{aligned} \tag{2.9}$$

and for the D3Q19 model,

$$\begin{aligned} w_0 &= \frac{1}{3} \\ w_{1-6} &= \frac{1}{18} \\ w_{7-17} &= \frac{1}{36}. \end{aligned} \tag{2.10}$$

2.4 Multiple-relaxation-time Lattice Boltzmann Method

The multiple-relaxation-time (MRT) LBM is an improvement to the LBGK model, where the collision operator is designed such that each moment in the set of distribution functions are relaxed at different rates. This allows the relaxation times for each moment to be

adjusted to optimize for stability. In this section, the MRT-LBM model for the D3Q19 lattice will be outlined [14].

The collision model for the MRT-LBM can be written as:

$$\boldsymbol{f}(\vec{r}+\vec{c}_i, t+1) = \boldsymbol{f}(\vec{r}, t) - \mathbf{M}^{-1} \cdot \mathbf{S} \cdot (\mathbf{M} \cdot \boldsymbol{f}(\vec{r}, t) - \mathbf{M} \cdot \boldsymbol{f}^{eq}(\vec{r}, t)), \tag{2.11}$$

where $\boldsymbol{f}$ denotes the distribution functions, and $\boldsymbol{f}^{eq}$ denotes the local equilibrium distribution functions. For this study, the D3Q19 lattice, where the discrete velocities are defined as:

$$\vec{c_i} = \begin{cases} (0,0,0), & i = 0 \\ (\pm 1,0,0),(0,\pm 1,0),(0,0,\pm 1), & i = 1,2,...,6 \\ (\pm 1,\pm 1,0),(\pm 1,0,\pm 1),(0,\pm 1,\pm 1), & i = 7,8,...,18, \end{cases} \tag{2.12}$$

was used (see also Figure 2.2).

The transformation matrix $\mathbf{M}$ is chosen such that the distribution functions are mapped to its moments, $\boldsymbol{m}$. For the D3Q19 lattice, it is given by:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -30 & -11 & -11 & -11 & -11 & -11 & -11 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & -4 & -4 & -4 & -4 & -4 & -4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -4 & 4 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & -4 & 4 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -4 & 4 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 0 & 2 & 2 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2 \\ 0 & -4 & -4 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 & -2 & -2 & -2 \\ 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \end{pmatrix}. \tag{2.13}$$

The corresponding moments for this matrix would be:

$$\boldsymbol{m} = (\rho, e, \varepsilon, j_x, q_x, j_y, q_y, j_z, q_z, 3p_{xx}, 3\pi_{xx}, p_{ww}, \pi_{ww}, p_{xy}, p_{yz}, p_{xz}, m_x, m_y, m_z)^T, \tag{2.14}$$

where ρ is the density, e and ε are related to the kinetic energy independent of the density, $\boldsymbol{j}$ is the momentum, $\boldsymbol{p}$ is the viscous stress tensor, $\boldsymbol{q}$ is the energy flux, $\boldsymbol{m}$ is the third-order moment tensor, and $\boldsymbol{\pi}$ is the fourth-order moment tensor. The equilibrium moments, $\boldsymbol{m}^{eq}$, for the non-conserved moments for the D3Q19 can be given as:

$$e^{eq} = -11\rho + \frac{19}{\rho_0}\boldsymbol{j}\cdot\boldsymbol{j}, \quad \varepsilon^{eq} = -\frac{475}{63}\boldsymbol{j}\cdot\boldsymbol{j} \tag{2.15}$$

$$q_x^{eq} = -\frac{2}{3}j_x, \quad q_y^{eq} = -\frac{2}{3}j_y, \quad q_z^{eq} = -\frac{2}{3}j_z \tag{2.16}$$

$$3p_{xx}^{eq} = \frac{1}{\rho_0}\left(2j_x^2 - (j_y^2 + j_z^2)\right), \quad p_{ww}^{eq} = \frac{1}{\rho_0}\left(j_y^2 - j_z^2\right) \tag{2.17}$$

$$p_{xy}^{eq} = \frac{1}{\rho_0}(j_x j_y), \quad p_{yz}^{eq} = \frac{1}{\rho_0}(j_y j_z), \quad p_{xz}^{eq} = \frac{1}{\rho_0}(j_x j_z) \tag{2.18}$$

$$3\pi_{xx}^{eq} = \pi_{ww}^{eq} = m_x^{eq} = m_y^{eq} = m_z^{eq} = 0. \tag{2.19}$$

The collision matrix $\mathbf{S}$ is a diagonal matrix consisting of:

$$\mathbf{S} = diag\left(0, s_1, s_2, 0, s_4, 0, s_4, 0, s_4, s_9, s_{10}, s_9, s_{10}, s_{13}, s_{13}, s_{13}, s_{16}, s_{16}, s_{16}\right), \tag{2.20}$$

where the zero elements correspond to the conserved moments in the LBM: mass and momentum. The molecular viscosity of the fluid is specified by s_9 and s_{13} as:

$$\nu = \frac{1}{3}\left(\frac{1}{s_9} - \frac{1}{2}\right) = \frac{1}{3}\left(\frac{1}{s_{13}} - \frac{1}{2}\right). \tag{2.21}$$

The other elements of $\mathbf{S}$ can be chosen arbitrarily, as the non-hydrodynamic moments do not directly affect the hydrodynamic solution. In this study, these elements were set to 1.

2.5 Recovering the Navier-Stokes Equations

In this section, the incompressible Navier-Stokes equations will be recovered from the LBE. For simplicity, the SRT LBE with the BGK collision operator will be used. The evolution equation for this model is:

$$f_i(\vec{r} + \vec{c_i}, t+1) = f_i(\vec{r}, t) - \frac{1}{\tau}(f_i - f_i^{eq}). \tag{2.22}$$

The incompressible Navier-Stokes equations can be recovered from this equation through the Chapman-Enskog analysis, which is essentially a formal multi-scaling expansion [11, 19]:

$$\frac{\partial}{\partial t} = \epsilon \frac{\partial}{\partial t_1} + \epsilon^2 \frac{\partial}{\partial t_2} + \cdots \tag{2.23}$$

$$\frac{\partial}{\partial x} = \epsilon \frac{\partial}{\partial x_1} + \cdots . \tag{2.24}$$

The expansion parameter, ϵ, is the Knudsen number, defined as the ratio between the mean free path of a gas molecule and a macroscopic length scale [31]. Using this technique, the particle distribution function, f_i, can be expanded about the local equilibrium function, f_i^{eq}, as:

$$f_i = f_i^{eq} + \epsilon f_i^{neq}. \tag{2.25}$$

Here, f_i^{eq} is defined from Equation 2.7, and must satisfy:

$$\sum_{i=0}^{q-1} f_i^{eq} = \rho \tag{2.26}$$

$$\frac{1}{\rho} \sum_{i=0}^{q-1} \vec{c}_i f_i^{eq} = \vec{u}. \tag{2.27}$$

The non-equilibrium distribution function, $f_i^{neq} = f_i^1 + \epsilon f_i^2 + \mathcal{O}(\epsilon^2)$, has the following constraints to ensure conservation of mass and momentum in the collision operator:

$$\sum_{i=0}^{q-1} f_i^k = 0 \tag{2.28}$$

$$\sum_{i=0}^{q-1} \vec{c}_i f_i^k = 0. \tag{2.29}$$

for both $k = 1$ and $k = 2$. Equation 2.5 can be Taylor expanded, and rewritten in the consecutive order of ϵ as [74]:

$$f_i^0 = f_i^{eq} \ : \ \mathcal{O}(\epsilon^0) \tag{2.30}$$

$$\left(\partial_{t_0} + \vec{c}_i \cdot \nabla\right) f_i^0 = -\frac{1}{\tau} f_i^1 \ : \ \mathcal{O}(\epsilon^1) \tag{2.31}$$

$$\partial_{t_1} f_i^0 + \left(\frac{2\tau - 1}{2\tau}\right) \left(\partial_{t_0} + \vec{c}_i \cdot \nabla\right) f_i^1 = -\frac{1}{\tau} f_i^2 \ : \ \mathcal{O}(\epsilon^2) \tag{2.32}$$

Using the constraints shown in Equations 2.26–2.29, and summing the first order expansion (Equation 2.31) for all i, the continuity equation can be obtained as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \vec{u} = 0. \tag{2.33}$$

Next, by multiplying $\vec{c_i}$ to both sides of Equations 2.31 and 2.32 and combining them, the momentum equation is derived as:

$$\frac{\partial (\rho \vec{u})}{\partial t} + \nabla \cdot \left(\mathbf{\Pi}^0 + \frac{2\tau - 1}{2\tau} \mathbf{\Pi}^1 \right) = 0, \tag{2.34}$$

where $\mathbf{\Pi}^0$ and $\mathbf{\Pi}^1$ are momentum flux tensors defined as:

$$\mathbf{\Pi}^0 = \sum_{i=0}^{q-1} \vec{c_i} \vec{c_i} f_i^0 = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta \tag{2.35}$$

$$\mathbf{\Pi}^1 = \sum_{i=0}^{q-1} \vec{c_i} \vec{c_i} f_i^1 = \nu \left(\nabla_\alpha (\rho u_\beta) + \nabla_\beta (\rho u_\alpha) \right). \tag{2.36}$$

p is the pressure, and is related to density as:

$$p = c_s^2 \rho = \frac{1}{3} \rho. \tag{2.37}$$

The resulting momentum equation becomes:

$$\rho \left(\frac{\partial \vec{u}}{\partial t} + \vec{u}.\nabla \vec{u} \right) = -\nabla p + \rho \nu \nabla^2 \vec{u}, \tag{2.38}$$

which is identical to the Navier-Stokes equations, given that the density fluctuations are small enough [11].

2.6 Boundary Conditions

At domain boundaries, or in the presence of solid walls, the general computational procedure outlined in the previous section encounters problems. After the streaming step, the particle distribution functions that are streamed from outside the domain are unknown 2.3. In the LBM, boundary conditions are implemented to model the unknown particle distributions.

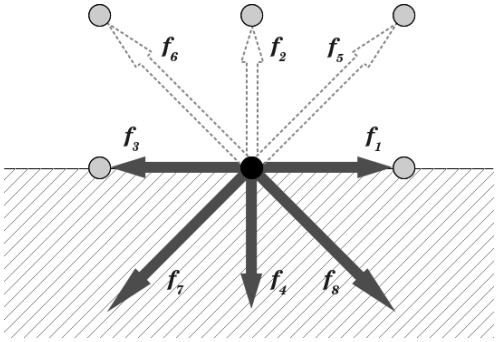

Figure 2.3: Particle distributions at the south boundary, after the streaming operation. Three particle distributions are "missing".

The simplest boundary condition that models a no-slip wall is referred to as the bounce back boundary. In this boundary condition, the incoming particle distributions are "bounced back", by changing their direction by 180°. For the south boundary in the D2Q9 lattice (Figure 2.3),

$$f_2 = f_4 \tag{2.39}$$
$$f_5 = f_7 \tag{2.40}$$
$$f_6 = f_8. \tag{2.41}$$

The collision step is skipped for these boundary nodes. It is important to note that the exact location of the boundary is located half way between the boundary node and the inner node. The bounce back boundary is trivial to implement, and is proved to have second order accuracy in the spatial discretization [71].

The symmetry boundary, or the free-slip wall, can be implemented in a similar fashion. Instead of copying the incoming distributions to the opposite direction, they are copied to its symmetric counterpart. For the south boundary in the D2Q9 lattice (Figure 2.3),

$$f_2 = f_4 \tag{2.42}$$
$$f_5 = f_8 \tag{2.43}$$
$$f_6 = f_7. \tag{2.44}$$

2.6.1 Velocity and Pressure Boundary Conditions

One of the major challenges in the LBM is the implementation of velocity and pressure boundary conditions. The difficulty stems from the fact that LBM relies on particle dis-

tribution functions instead of the hydrodynamic variables, which leads to a higher degree of freedom on a per node basis. This means that specifying the pressure and velocities at a given node does not completely define the state of a node. Hence, there is a need to introduce some assumptions to recover the particle distributions from a given hydrodynamic state. Here, the method proposed by Mussa *et al.* [41] is outlined.

For a 2D computational domain with a Cartesian grid, let the nodes be indexed by (i, j), where $i \in \{1, 2, ..., N_x\}$ and $j \in \{1, 2, ..., N_y\}$. In order to impose a velocity boundary condition at $i = 1$, the velocity at the nodes of $i = 1$ are set to the prescribed velocity, $\boldsymbol{u}$. The remaining moments, $\boldsymbol{m}$, at the boundary are simply copied from the nodes at $i = 2$. Since all of the moments on the boundary are now known, the moments can be transformed into distribution functions, $\boldsymbol{f}$. This procedure can be written as:

$$\boldsymbol{f}(i = 1, j) = \mathbf{M}^{-1} \cdot \boldsymbol{m}^*(i = 2, j)|_{\boldsymbol{u}=u_{in}}, \tag{2.45}$$

where $\boldsymbol{m}^*$ refers to the post-collision moments. Pressure boundary conditions can be implemented in a similar fashion, as:

$$\boldsymbol{f}(i = 1, j) = \mathbf{M}^{-1} \cdot \boldsymbol{m}^*(i = 2, j)|_{\boldsymbol{\rho}=\rho_{in}}. \tag{2.46}$$

Since this method only consists of copying over the moments from its neighbor node, it is trivial to implement in a 3D domain as well.

2.6.2 Curved Boundary Treatments

The most widely used boundary condition for no-slip walls is the bounce back (BB) scheme, where the particle distributions are simply reflected at the solid node. For the BB scheme, the exact location of the wall is known to be half way between the solid node and its adjacent fluid node [22]. This method is often sufficient to model straight walls, but in order to accurately model the curved boundaries on the cylinder surface, additional treatments are required. For this study, the interpolated bounce back (IBB) scheme [7] was adopted. This method utilizes interpolations to estimate the particle distribution function at a location that is coherent with the wall location. The details of this method are explained as follows.

Let q be the location of the wall, measured from the fluid node adjacent to the wall (Figure 2.4). For the BB scheme, the wall is located at $q = 0.5$, which means that the particles traverse a total distance of δ. Assuming $\delta = 1$, for $q < 0.5$, a fictitious node, r_B, is assumed, such that

$$\|r_A - r_B\| = 1 - 2q. \tag{2.47}$$

The particles departing r_B should travel a distance of 1 as it collides with the wall, and reach r_A. The distribution function at r_B can be estimated by a linear interpolation between r_A and $r_A - c_i$, hence yielding:

$$\begin{aligned} f_{i'}(r_A, t+1) &= f_i^*(r_B, t) \\ &= 2q f_i^*(r_A, t) + (1-2q) f_i^*(r_A - c_i, t), \end{aligned} \tag{2.48}$$

where i' denotes the opposite velocity of i (i.e. $c_{i'} = -c_i$), and f^* denotes the post-collision distribution.

In the case where $q \geq 0.5$, the f_i particles leaving r_A will collide with the wall, and arrive at a fictitious node, r_B, where

$$\|r_A - r_B\| = 2q - 1. \tag{2.49}$$

At the same time, the $f_{i'}$ particles from r_A will reach node $r_A - c_i$. Therefore, $f_{i'}$ at r_A for the next time step can be obtained as:

$$\begin{aligned} f_{i'}(r_A, t+1) &= \frac{(2q-1)}{2q} f_{i'}(r_A - c_i, t+1) + \frac{1}{2q} f_{i'}(r_B, t+1) \\ &= \frac{(2q-1)}{2q} f_{i'}^*(r_A, t) + \frac{1}{2q} f_i^*(r_A, t). \end{aligned} \tag{2.50}$$

Note that it is also possible to use quadratic interpolations to improve the accuracy of this method. In this study, linear interpolations were used for its simplicity.

2.6.3 Force Evaluations

The forces experienced by the solid boundary can be obtained by computing the change in momentum of the particle distribution functions. The momentum transfer at a given boundary link can be shown to be [30]:

$$\delta F_i(r_A + c_i, t + \delta t/2) = c_i \left[f_i^*(r_A, t) + f_{i'}(r_A + c_i, t) \right]. \tag{2.51}$$

By summing the forces from each boundary link, it is possible to deduce the total hydrodynamic force exerted on the solid boundary.

2.7 Local Grid Refinement

Incorporating local grid refinements is a very effective way of reducing the computational cost of grid based numerical methods. In the current work, blocks of refined grids were

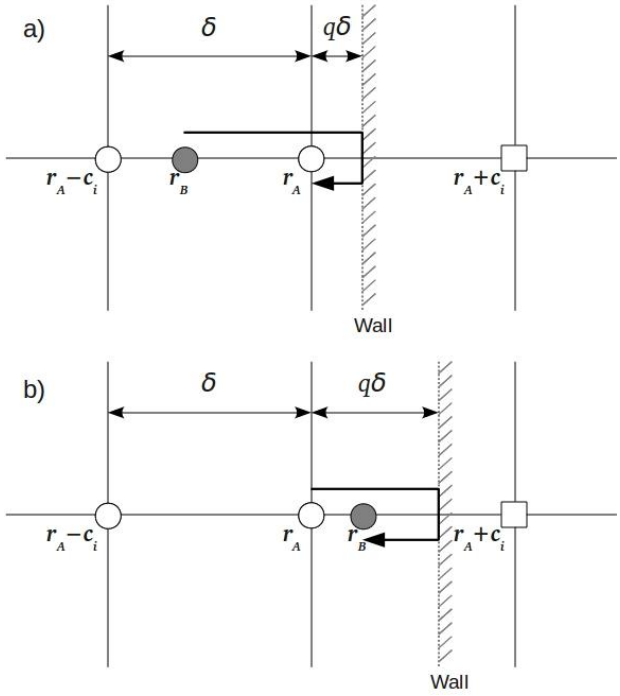

Figure 2.4: Schematic for the interpolated bounce back method. (a) $q < 0.5$. (b) $q \geq 0.5$.

overlaid on top of the base grid in a hierarchical fashion, using the method proposed by Filippova and Hänel [17]. In this method, the lattice speed is kept constant, while the grid size, δ_x, and time step size, δt, are reduced by the refinement factor, n. Since $\nu = (1/s_9 - 1/2)\delta_x c/3$, in order to achieve the same viscosity as the base grid, the relaxation parameter, s_9, must be adjusted as:

$$s_{9,f} = \frac{2}{1 + n(2/s_{9,c} - 1)}, \tag{2.52}$$

where the subscript f and c refer to the fine and coarse grids, respectively. Furthermore, to keep the hydrodynamic variables and their derivatives continuous between the fine and coarse grids, the non-equilibrium portion of the post-collision distribution functions (or alternatively its moments) must scale according to:

$$\boldsymbol{f}_c^* = \boldsymbol{f}_f^{eq} + \left(\boldsymbol{f}_f^* - \boldsymbol{f}_f^{eq}\right) \frac{(1 - s_{9,c}) s_{9,f} n}{s_{9,c}(1 - s_{9,f})} \tag{2.53}$$

$$\boldsymbol{f}_f^* = \boldsymbol{f}_c^{eq} + \left(\boldsymbol{f}_c^* - \boldsymbol{f}_c^{eq}\right) \frac{s_{9,c}(1 - s_{9,f})}{(1 - s_{9,c}) s_{9,f} n}. \tag{2.54}$$

Note that by scaling the distribution functions in phase space instead of moment space, it is implied that all moments are scaled based on the value of s_9. By choosing to perform the scaling in moment space, one can scale each moment based on its corresponding element in $\mathbf{S}$.

In this study, the refined grid is oriented such that the fine grid nodes are all offset from the coarse grid nodes, and n is set to 2 for each refinement (Figure 2.5). Since the time step size is halved for each refinement level, the refined grid must be marched by two time steps for each time step in the coarse grid. After each streaming step in the fine grid, the nodes on the outer edge of the fine grid (indicated by the red nodes in Figure 2.5) will have unknown particle distributions. For such nodes, the particle distributions are interpolated from the coarse grid both spatially and temporally. Spatial interpolations can be performed by using bilinear (2D) or trilinear (3D) interpolations with its closest neighbors, and linear interpolations can be used for the temporal interpolations [18]. For example, the distribution functions at node a in Figure 2.5 for $t = t_0$ can be obtained as:

$$\boldsymbol{f}_{t_0}^a = \frac{1}{16}\left(9\boldsymbol{f}_{t_0}^D + 3\boldsymbol{f}_{t_0}^B + 3\boldsymbol{f}_{t_0}^C + \boldsymbol{f}_{t_0}^A\right), \tag{2.55}$$

and for $t = t_0 + \frac{1}{2}$, the coarse grid solution at $t_0 + 1$ is spatially interpolated to obtain $\boldsymbol{f}_{t_0+1}^a$, which is used for the linear interpolation in time:

$$\boldsymbol{f}_{t_0+\frac{1}{2}}^a = \frac{1}{2}\left(\boldsymbol{f}_{t_0}^a + \boldsymbol{f}_{t_0+1}^a\right). \tag{2.56}$$

At $t = t_0 + 1$, the distributions on the fine grid are spatially averaged, and copied onto the corresponding coarse grid nodes.

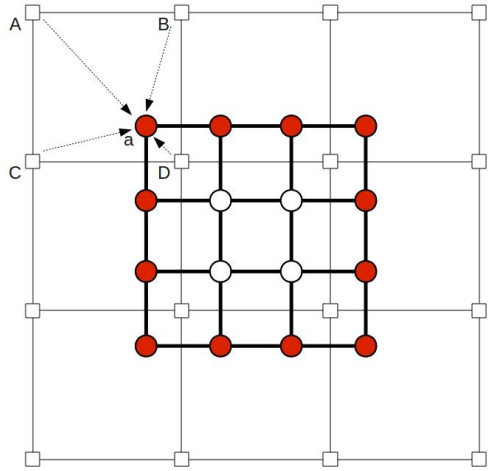

Figure 2.5: Schematic of local grid refinement by a factor of 2. The fine grid nodes are offset from the coarse grid nodes. Red nodes indicate nodes that require spatial and temporal interpolations.

For the MRT LBM, it is also possible to employ a quadratic bubble function to realize a second order interpolation in space [20, 66]. In this method, the distributions at the neighboring nodes are converted into moments, interpolated using a compact second order scheme, and converted back into particle distributions. The method can be outlined as follows.

First, consider the cell consisting of the four nodes, A, B, C, and D, in Figure 2.5, and let the coordinates of each node be denoted as:

$$A : (x_0, y_0 + h) \tag{2.57}$$
$$B : (x_0 + h, y_0 + h) \tag{2.58}$$
$$C : (x_0, y_0) \tag{2.59}$$
$$D : (x_0 + h, y_0). \tag{2.60}$$

A velocity field of the following form is assumed.

$$u_x(x', y') = a_0 + a_1 x' + a_2 y' + a_3 x' y' + c_x(1 - x'^2) + c_y(1 - y'^2) \tag{2.61}$$

$$u_y(x', y') = b_0 + b_1 x' + b_2 y' + b_3 x' y' + d_x(1 - x'^2) + d_y(1 - y'^2) \tag{2.62}$$

where

$$x' = \frac{2(x - x_0)}{h} - 1 \tag{2.63}$$

$$y' = \frac{2(y - y_0)}{h} - 1. \tag{2.64}$$

The coefficients a and b relate to the bilinear interpolation, while the coefficients c and d relate to the second order quadratic interpolation. The values for a and b can be obtained by enforcing the known velocities at the four coarse grid nodes. For u_x, the constraints give:

$$\begin{pmatrix} u_x^A \\ u_x^B \\ u_x^C \\ u_x^D \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}. \tag{2.65}$$

Inverting the matrix leads to:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} u_x^A \\ u_x^B \\ u_x^C \\ u_x^D \end{pmatrix}. \tag{2.66}$$

Similarly, for u_y,

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} u_y^A \\ u_y^B \\ u_y^C \\ u_y^D \end{pmatrix}. \tag{2.67}$$

To compute the quadratic coefficients, c and d, information on the elements of the strain rate tensor are computed at the coarse grid nodes. In the MRT LBM model, the strain rate elements are related to the non-equilibrium moments as [66]:

$$\partial_x u_x = S_{xx} = \frac{s_9}{c^2 \delta t} \left[\frac{3}{4} \left(u_x^2 - u_y^2 - \frac{p_{xx}}{\rho_0} \right) \right] + \mathcal{O}(h^2) \tag{2.68}$$

$$\partial_y u_y = S_{yy} = \frac{s_9}{c^2 \delta t} \left[\frac{3}{4} \left(u_y^2 - u_x^2 + \frac{p_{xx}}{\rho_0} \right) \right] + \mathcal{O}(h^2) \tag{2.69}$$

$$\partial_x u_y + \partial_y u_x = 2S_{xy} = 3\frac{s_9}{c^2 \delta t} \left[\frac{3}{4} \left(u_x u_y - \frac{p_{xy}}{\rho_0} \right) \right] + \mathcal{O}(h^2) \tag{2.70}$$

At each of the four nodes, A, B, C, and D, Equations 2.68 to 2.70 are applied as constraints, which leads to a system of 12 equations:

$$\begin{pmatrix} 4/h & 0 & 0 & 0 \\ 0 & 0 & 4/h & 0 \\ 0 & 4/h & 0 & 4/h \\ -4/h & 0 & 0 & 0 \\ 0 & 0 & 4/h & 0 \\ 0 & 4/h & 0 & -4/h \\ -4/h & 0 & 0 & 0 \\ 0 & 0 & -4/h & 0 \\ 0 & -4/h & 0 & -4/h \\ 4/h & 0 & 0 & 0 \\ 0 & 0 & -4/h & 0 \\ 0 & -4/h & 0 & 4/h \end{pmatrix} \cdot \begin{pmatrix} c_x^A \\ c_y^B \\ d_y^C \\ d_y^D \end{pmatrix} = \begin{pmatrix} \frac{u_x^C - u_x^D}{h} + S_{xx}^C \\ \frac{u_y^C - u_y^A}{h} + S_{yy}^C \\ \frac{u_x^C - u_x^A + u_y^C - u_y^D}{h} + 2S_{xy}^C \\ \frac{u_x^C - u_x^D}{h} + S_{xx}^D \\ \frac{u_y^D - u_y^B}{h} + S_{yy}^D \\ \frac{u_x^D - u_x^B + u_y^C - u_y^D}{h} + 2S_{xy}^D \\ \frac{u_x^A - u_x^B}{h} + S_{xx}^B \\ \frac{u_y^D - u_y^B}{h} + S_{yy}^B \\ \frac{u_x^D - u_x^B + u_y^A - u_y^B}{h} + 2S_{xy}^B \\ \frac{u_x^A - u_x^B}{h} + S_{xx}^A \\ \frac{u_y^C - u_y^A}{h} + S_{yy}^A \\ \frac{u_x^C - u_x^A + u_y^A - u_y^B}{h} + 2S_{xy}^A \end{pmatrix}. \tag{2.71}$$

Of the 12 equations, only four are linearly independent, the original authors for this interpolation method used the Moore-Penrose pseudo-inverse to compute the coefficients, c_x, c_y, d_y, d_x [66]. It is also possible to simply combine the linearly dependent equations to reduce the system to four equations. The resulting expressions for the quadratic coefficients are:

$$c_x = \frac{h}{16}[S_{xx}^A + S_{xx}^C - S_{xx}^B - S_{xx}^D] \tag{2.72}$$

$$d_y = \frac{h}{16}[S_{yy}^C + S_{yy}^D - S_{yy}^A - S_{yy}^B] \tag{2.73}$$

$$c_y = \frac{h}{8}[S_{xy}^C + S_{xy}^D - S_{xy}^A - S_{xy}^B] + \frac{h}{8}[u_y^C + u_y^B - u_y^A - u_y^D] \tag{2.74}$$

$$d_x = \frac{h}{8}[S_{xy}^C + S_{xy}^A - S_{xy}^D - S_{xy}^B] + \frac{h}{8}[u_x^C + u_x^B - u_x^A - u_x^D]. \tag{2.75}$$

Hence, by utilizing the information on the elements of the strain rate tensor, the LBM allows for a compact, second order interpolation for the velocity field. The other moments, such as density and strain rates, can be obtained from a simple bilinear (trilinear) interpolation. Once all of the moments are known, the distribution functions can be recovered as:

$$\boldsymbol{f} = \mathbf{M}^{-1} \cdot \boldsymbol{m}. \tag{2.76}$$

2.8 Smagorinsky Subgrid Model for the Lattice Boltzmann Method

In order to model the unresolved scales of motion at high Reynolds numbers, subgrid models are often employed. For LES, the subgrid models are applied after a spatial filtering operation, defined as [51]:

$$\overline{w}(x) = \int w(x)G(x,x')dx' \tag{2.77}$$

where w is a spatially dependent quantity, and G the kernel function. Based on this filtering function, the filtered particle distribution function, $\overline{f}_i$, can be defined as:

$$\overline{f}_i(x) = \int f_i(x)G(x,x')dx'. \tag{2.78}$$

Thus, the filtered LBE becomes:

$$\overline{f}_i(\vec{r}+\vec{c_i},t+1) = \overline{f}_i(\vec{r},t) + \overline{\Omega}_i. \tag{2.79}$$

From here, it is assumed that the filtered particle distribution will relax towards a local filtered equilibrium distribution, which only depends on the local filtered macroscopic variables ($\overline{\rho}$ and $\overline{\vec{u}}$) [25]. Hence,

$$\overline{\Omega_i(f(\vec{r},t))} = \Omega_i(\overline{f(\vec{r},t)}), \tag{2.80}$$

where the equilibrium distribution function, $\overline{f}_{eq}$, is:

$$\overline{f}_i^{eq} = \overline{\rho}t_i\left(1 + \frac{1}{c_s^2}\vec{c_i}\cdot\overline{\vec{u}} + \frac{1}{2c_s^2}\overline{\boldsymbol{Q}}_i : \overline{\vec{u}\vec{u}}\right). \tag{2.81}$$

For subgrid closure, the Smagorinsky model [60], which relates the eddy viscosity to the local strain rate tensor, was used. The governing equation for this model is:

$$\nu_{total} = \nu_0 + C\Delta^2\left|\overline{\mathbf{S}}\right|, \tag{2.82}$$

where ν_{total} is the total effective viscosity, ν_0 is the molecular viscosity, Δ is the filter size, C is the Smagorinsky constant, and $\left|\overline{\mathbf{S}}\right|$ is related to the magnitude of the filtered strain rate tensor as:

$$\left|\overline{\mathbf{S}}\right| = \sqrt{\mathbf{2}\overline{\mathbf{S}} : \overline{\mathbf{S}}}, \tag{2.83}$$

where

$$\overline{\mathbf{S}} = \frac{1}{2}(\nabla \overline{u} + (\nabla \overline{u})^{\top}). \tag{2.84}$$

In the LBM, the viscosity of the fluid is governed by the relaxation time. Hence, the Smagorinsky subgrid model can be implemented by locally adjusting the relaxation time in the LBE. From Equation 2.6,

$$\nu_{total} = c\delta x \frac{2\tau_{total} - 1}{6}. \tag{2.85}$$

Combining this with Equation 2.82,

$$\begin{aligned} \tau_{total} &= \frac{3}{c\delta x}\left(\nu_0 + C\Delta^2 \left|\overline{\mathbf{S}}\right|\right) + \frac{1}{2} \\ &= \tau_0 + \frac{3}{c\delta x} C\Delta^2 \left|\overline{\mathbf{S}}\right|, \end{aligned} \tag{2.86}$$

where τ_0 is the relaxation time obtained from the molecular viscosity. The filtered strain rate tensor can be computed directly from the non-equilibrium momentum flux tensor, $\mathbf{\Pi^1}$ [25]. Using Equation 2.83, it can be seen that Equation 2.36 can be written as:

$$\overline{\mathbf{\Pi}}^1 = \sum_{i=0}^{q-1} \vec{c}_i \vec{c}_i \overline{f}_i^1 = -\frac{2\overline{\rho}\tau_{total} c\delta x}{3} \overline{\mathbf{S}}. \tag{2.87}$$

Taking the tensorial magnitude, and defining $Q^{1/2} = \sqrt{\overline{\mathbf{\Pi}}^1 : \overline{\mathbf{\Pi}}^1}$,

$$Q^{1/2} = \frac{\sqrt{2}\tau_{total}\overline{\rho} c\delta x}{3} \left|\overline{\mathbf{S}}\right| \tag{2.88}$$

Substituting Equation 2.86 into 2.88 leads to a quadratic equation for $\left|\overline{\mathbf{S}}\right|$, which can be solved to obtain:

$$\left|\overline{\mathbf{S}}\right| = \frac{-\tau_0 \overline{\rho} c\delta x + \sqrt{(\tau_0 \overline{\rho} c\delta x)^2 + 18\sqrt{2}\overline{\rho} C\Delta^2 Q^{1/2}}}{6\overline{\rho} C\Delta^2}. \tag{2.89}$$

Substituting Equation 2.89 into Equation 2.86 and assuming $\delta x = \Delta$ (implicit filtering), the final expression for τ becomes:

$$\tau_{total} = \frac{\tau_0}{2} + \frac{\sqrt{(\tau_0 \overline{\rho} c)^2 + 18\sqrt{2}\overline{\rho} C Q^{1/2}}}{2\overline{\rho} c}. \tag{2.90}$$

Contrary to the finite volume Navier-Stokes solvers that require finite difference schemes to compute $\left|\overline{\mathbf{S}}\right|$, the LBM allows direct computation of $\left|\overline{\mathbf{S}}\right|$ using local variables.

2.9 Computational Procedure

For computational implementation, the LBE (Equation 2.2) is often split into two steps: the collision step, and the streaming step. The collision step involves updating the local particle distribution functions according to the collision operator, Ω_i, as:

$$\tilde{f}_i(\vec{r}, t) = f_i(\vec{r}, t) + \Omega_i, \tag{2.91}$$

where $\tilde{f}_i$ represents the post-collision particle distribution. In order to realize the collision process, the hydrodynamic variables, ρ and $\vec{u}$, are obtained from Equations 2.3 and 2.4. For the SRT model, Equation 2.7 is then used to compute the equilibrium distributions, and Equation 2.5 is used to complete the collision step. In the streaming step, $\tilde{f}_i$ is moved along the discretized velocity space, as:

$$f_i(\vec{r} + \vec{c_i}, t + 1) = \tilde{f}_i(\vec{r}, t). \tag{2.92}$$

Completing the two steps constitutes one time step of the simulation. It should be noted that the procedure only involves information local to the node, or from its neighboring nodes, thus making the LBM a highly parallel efficient method.

Chapter 3

3D Simulation of the Laminar Flow Over Two Tandem Cylinders

3.1 Problem Description

In this chapter, the validity of the LBM code is evaluated in the context of simulating the laminar flow over bluff bodies. A classical case for this type of flow is the flow over a circular cylinder, which has been studied extensively for over 100 years, as it contains a wide range of complex flow phenomena, despite its simple geometry. It is known that the flow displays characteristic regimes based on its Reynolds number (Re), the most famous one being the formation of the von Kármán vortex street, where counter-rotating vortices are shed from the cylinder in a periodic fashion. This two-dimensional flow is known to transition to three-dimensional flow at $Re \approx 190$, as suggested by numerical and experimental work [5, 63, 69]. This transition is referred to as the introduction of Mode A instabilities, which are characterized by the span-wise periodic deformation of the primary von Kármán vortices. There is also a subsequent stage at $Re > 250$, where finer stream-wise vortex structures appear, which is referred to as the Mode B instabilities. More details on studies involving the wake dynamics of cylinders can be found in the review by Williamson [70].

This class of flow evolves in complexity when an additional cylinder is placed in the wake of the other. The tandem configuration of the cylinders involves wake interaction that is strongly dependent on the Reynolds number and the spacing between the cylinders. A review paper by Sumner that focuses on two cylinders in cross flow can be found in [62]. The general flow structure can be categorized based on the cylinder spacing, s. Here, the

flow classification by Carmo *et al.* [10] is adopted. For small s ($\sim 1.5d$), the shear layers separate from the upstream cylinder, and reattach onto the downstream cylinder, forming a nearly steady flow between the cylinders. This stage is referred to as SG (symmetric in the gap). For $s \sim 2d - 3d$, vortices are not shed in the interstitial region, but vortices start to grow and decay alternately, hence named the AG (alternating in the gap) regime. Further increasing the spacing past the critical spacing, s_c, allows the vortex street to form between the cylinders, causing the force coefficients and the Strouhal number (St) to rapidly increase. This final regime is referred to as WG (wake in the gap).

In the past, a number of numerical studies have been conducted in 2D for this flow at low Reynolds numbers [26, 32, 36, 38, 39, 58], but there are very few studies that focus on the 3D effects. Papaioannou *et al.* [50] conducted 2D and 3D direct numerical simulations with two tandem cylinders at $100 \leq Re \leq 1000$, with spacings ranging from $1.1d$ to $5d$. They found that 2D simulations tend to under-predict the critical spacing, and that for $s < s_c$, the 3D effects are suppressed, while for $s > s_c$, the 3D effects generally increase. Deng *et al.* [13] investigated the effects of the cylinder spacing and Reynolds number on the three dimensionality of the flow. This was done by varying the spacing from $1.5d$ to $8d$ at $Re = 220$, and varying the Re from 220 to 270 at a spacing of $3.5d$. It was found that at $Re = 220$, the wake is 3D for $s \geq 4d$, and at $s = 3.5d$, the Mode A instabilities appear at $Re = 250$. Carmo *et al.* [8] used the spectral element method in 2D and 3D for $1.5 \leq s \leq 8$ at $160 \leq Re \leq 320$, and analyzed the 3D instabilities with a focus on formation length and St. They concluded that when $Re > 190$, 2D simulations cannot accurately predict the critical spacing. This study was followed up in 2010, where Carmo *et al.* [10] conducted a detailed analysis on the secondary instabilities involved in this flow using a spectral method based direct numerical simulation together with asymptotic stability theory. Their study confirmed the findings reported in [8, 13, 50], and elucidated the specific critical Re and spacing for each unstable mode for the different regimes in the flow. They also present another study [9] that focuses on the hysteresis in the transition between different flow regimes and its relation with the conditions for 3D instabilities.

These past studies have explicated the conditions and mechanisms responsible for the development of 3D instabilities in the flow past tandem cylinders with great detail. However, to the author's knowledge, there are no studies that focus on the consequences of the inception of 3D flow structures in terms of aerodynamic forces. For many practical applications, the forces exerted by the flow field is of major concern, and accurately predicting these values is critical for engineering design.

The objective of this chapter is to investigate the effects of the early span-wise instabilities on the aerodynamic forces experienced by the cylinders. The Re of interest are at $160 \leq Re \leq 220$, which is where the Mode A instabilities are expected to develop. Both 2D and

3D simulations were conducted at non-dimensional cylinder spacings of 1.5, 3.0, 3.5, 4.0, 5.0, and 8.0, each at Reynolds numbers of 160, 180, 200, and 220. Simulations were also performed on single cylinders, and the results were summarized in terms of St and span-wise averaged force coefficients. It should be stressed that the objective of the simulations was not to accurately predict the critical Re for transition to 3D flow, but rather to quantify the aerodynamic consequences from the presence of the span-wise modes. The results presented in this chapter have been published as a journal article [28].

3.2 Validation

3.2.1 Single Cylinder at Re=100

As a preliminary validation study, the well known 2D incompressible flow over a single circular cylinder in an unbounded domain was simulated. For the 2D simulations, the 3D code was used with a span-wise dimension of 1 lattice unit with span-wise periodicity, to ensure consistency when comparing 2D and 3D results. Since the refinement technique employed for this study introduces refinements in the span-wise direction, the mesh for the 2D simulations have a maximum of 4 nodes in the span-wise direction, so it is not a true 2D simulation in the strictest sense. However, the span-wise dimension is only 1/5 of the diameter of the cylinder, which is expected to be small enough to suppress the development of 3D flow structures.

The Reynolds number defined as $Re \equiv U_\infty d/\nu$, where U_∞ is the free stream velocity, d is the diameter of the cylinder, and ν is the kinematic viscosity, was set to 100. At this Reynolds number, the flow is considered to be truly 2D. The domain geometry for this case is shown in Figure 3.1. All dimensions presented are normalized by d, which was set to be 5 lattice units in the base mesh. Local mesh refinements were applied near the cylinder to enhance the accuracy of the simulation. The initial condition was set to be at a uniform flow of $u_x = 0.08c$, where c is the lattice speed (normally set to 1), and a small perturbation was introduced to accelerate the formation of the vortex street. Computation was carried on until a constancy of $\pm 1\%$ was achieved in its mean and RMS lift and drag coefficients, which typically required $\sim$ 200 time units ($\tilde{t} = U_\infty t/d$) from the initial condition, and statistics were collected for at least 300 time units.

The moment extrapolation method [41] was used to set the inlet velocity to $u_x = 0.08c$ and $u_y = 0$, and the outlet density to 1.0. The IBB [7] was used to model the no-slip wall boundaries on the cylinder surface. Symmetry boundaries were applied to the top and bottom walls by simply reflecting the particle distributions about the $x - z$ plane.

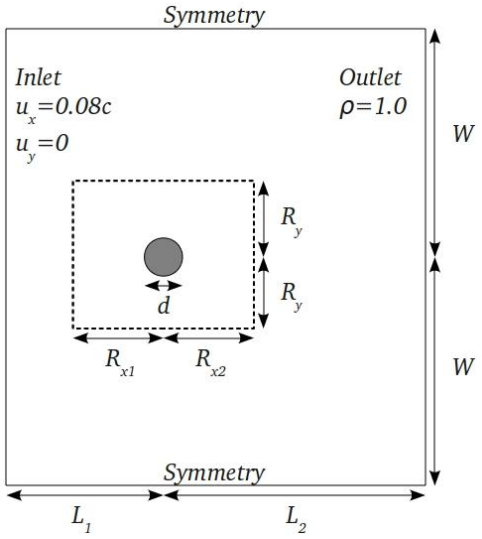

Figure 3.1: Domain geometry and boundary conditions for a single cylinder at Re=100. Mesh refinement region shown in dashed line.

A reference case was performed with $L_1 = 10d$, $L_2 = 15d$, $W = 20d$, with local refinements spanning $\pm 3d$ in the stream-wise direction and $\pm 2d$ in the cross-stream direction from the cylinder. Mesh refinement was achieved by overlaying two grids, each with a refinement factor of 2, which results in having $5 \times 2^2 = 20$ nodes along the diameter of the cylinder. The domain dimensions L_1, L_2, W, the local refinement region, and the local refinement level, were then adjusted to larger values one at a time, to assess the validity of the setup.

The results obtained from the base setup showed good agreement with the Navier-Stokes solver results by Sharman *et al.* [58], as well as the LBM results by Shu *et al.* [59] (Table 3.1). The table also shows that increasing the domain geometry or further refining the mesh only contributes to marginal improvements in the results, so for the following simulations, the dimensions from the reference case were chosen. The inlet velocity had some notable effect on the obtained results, which was expected, since it is known that the accuracy of the LBM at the incompressible limit is dependent on the Mach number [53, 21]. This dependency stems from the fact that LBM is a pseudo-compressible method applied to incompressible flow, much like the artificial compressibility method [2, 12, 44]. A higher order solution for $\overline{C}_D$ based on u_x, estimated by Richardson extrapolation was found to be 1.404. This suggests that inlet velocities of $u_x = 0.08c$, $u_x = 0.04c$, and $u_x = 0.02c$ result in relative errors of 2.77%, 0.99%, and 0.35%, respectively. Although the errors decrease more than linearly with only a linear increase in computational time, a lower u_x

Table 3.1: Numerical results for flow over circular cylinder at Re=100

	$\overline{C}_D$	$C_L{}'$	St
Reference case	1.365	0.238	0.161
Inlet distance $L_1 = 12d$	1.352	0.237	0.160
Outlet distance $L_2 = 20d$	1.367	0.240	0.162
Side wall distance $W = 25d$	1.365	0.239	0.161
Refinement region (downstream) $R_{x2} = 5d$	1.368	0.240	0.161
Refinement region (cross-stream) $R_y = 3d$	1.367	0.238	0.161
Refinement level $\times 2^3$	1.344	0.232	0.163
Inlet Velocity $u_x = 0.04c$ $(Ma = 0.069)$	1.390	0.245	0.162
Inlet Velocity $u_x = 0.02c$ $(Ma = 0.035)$	1.399	0.247	0.163
Shu *et al.* [59]	1.383	0.247	0.161
Sharman *et al.* [58]	1.33	0.23	0.164

leads to a higher relaxation rate, which makes the simulation more susceptible to numerical instabilities. Therefore, for the following studies, u_x was kept at $0.08c$.

3.2.2 Two Cylinders in Tandem at Re=100

The LBM code was further validated by simulating the 2D flow over two cylinders in tandem at Re=100. The problem geometry is shown in Figure 3.2. L_1, L_2, W, and the boundary conditions were kept the same as the single cylinder case. The local refinement region spanned $3d$ upstream of the first cylinder to $3d$ downstream of the second cylinder in the stream-wise direction, and $\pm 2d$ in the cross-stream direction from the cylinder center line. Since it is known that the flow regime transitions display hysteresis [39], care was taken so that the flow develops from an initially quiescent state. The same convergence criterion was used as the single cylinder case. Simulations were conducted with cylinder spacings, s, of $1.5d$, $3d$, $3.5d$, $4d$, $5d$, and $8d$. Figures 3.4 to 3.6 summarize the St as well as the mean and RMS values of the lift and drag coefficients for both cylinders.

The critical spacing was found to be at $3.5d < s_c < 4d$, and the results were generally consistent with the LBM results presented by Mussa *et al.* [41] and Navier-Stokes solver results by Sharman *et al.* [58]. Hence, it was reasonable to conclude that the LBM code used for this study is capable of capturing the general flow physics involving two cylinders in tandem.

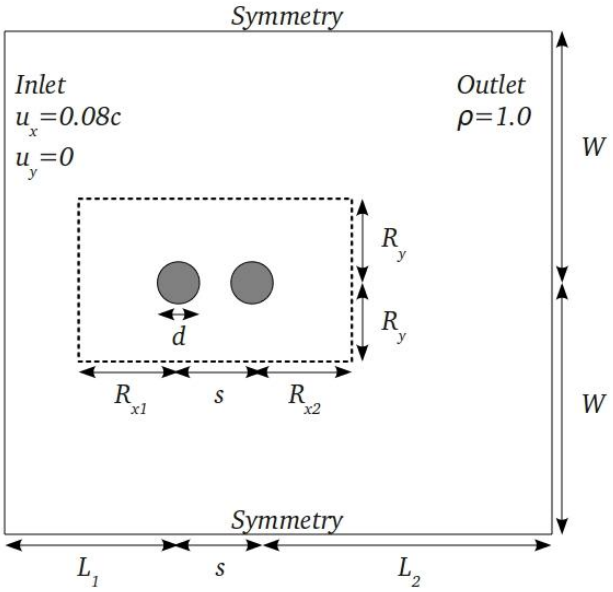

Figure 3.2: Domain geometry and boundary conditions for two cylinders in tandem at Re=100. Mesh refinement region shown in dashed line.

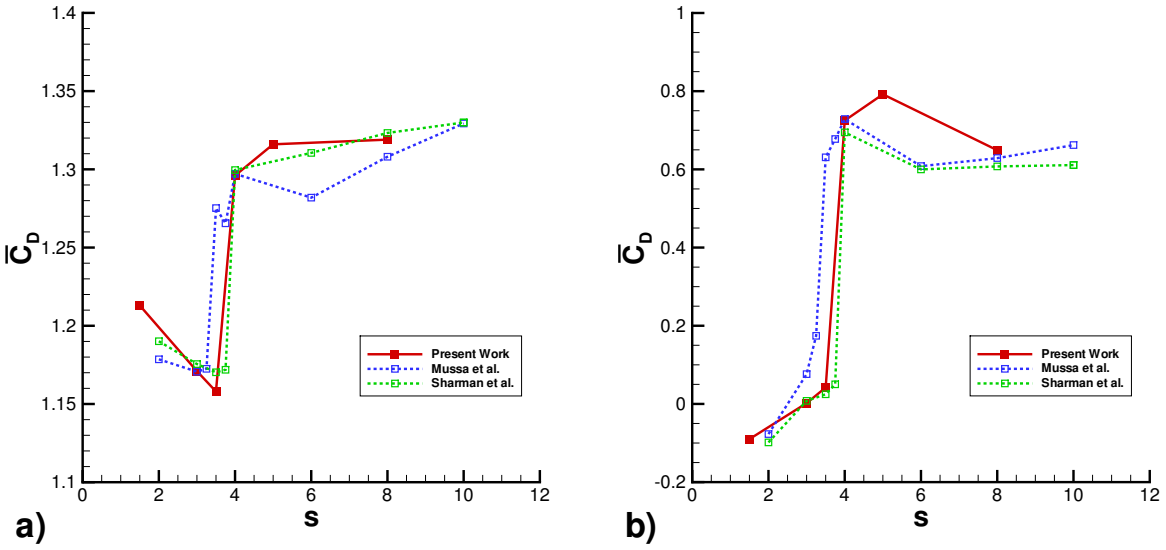

Figure 3.3: $\overline{C}_D$ against cylinder spacing, at Re=100. a) upstream cylinder; b) downstream cylinder.

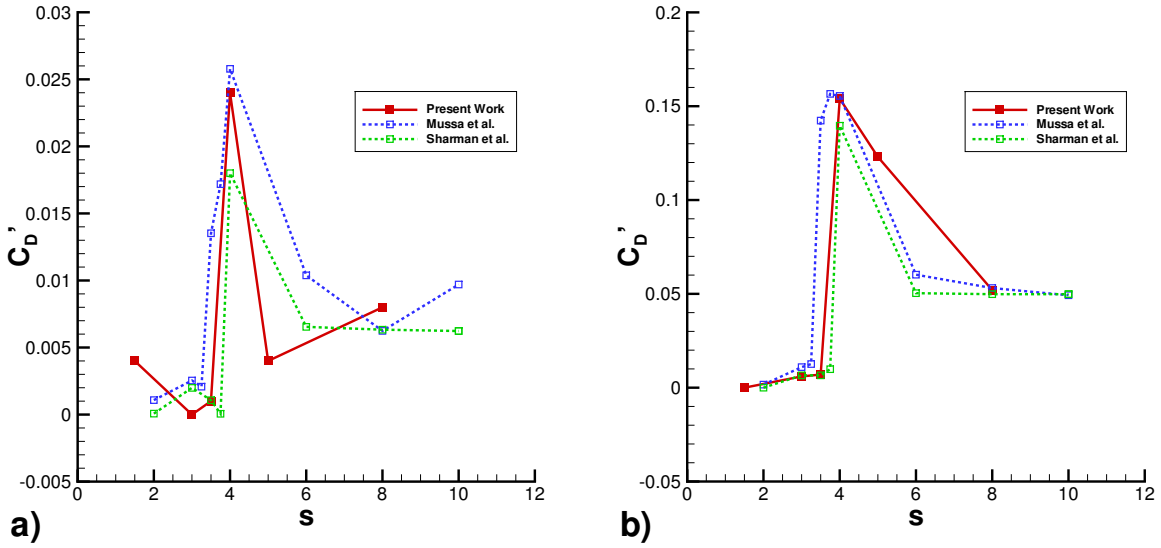

Figure 3.4: C'_D against cylinder spacing, at Re=100. a) upstream cylinder; b) downstream cylinder.

3.3 2D and 3D Simulations at $160 \leq Re \leq 220$

3.3.1 Mean and RMS Force Coefficients and Strouhal Number in 2D

2D and 3D simulations were conducted on one and two cylinders in tandem at Reynolds numbers of 160, 180, 200, and 220, at cylinder spacings of $1.5d$, $3d$, $3.5d$, $4d$, $5d$, and $8d$. The simulation setup was made identical to that used in the validation study with two cylinders. Computations were carried out until its statistics reached a constancy of $\pm 1\%$, and at least 25 shedding cycles were recorded to extract statistical data. The results are summarized in Table 3.2.

For the Reynolds numbers investigated, the critical spacing was $3.5d < s_c < 4d$. The effects of changing the spacing, s, were consistent for all Re, and there was no evidence of any change in the general flow phenomenon with increased Re.

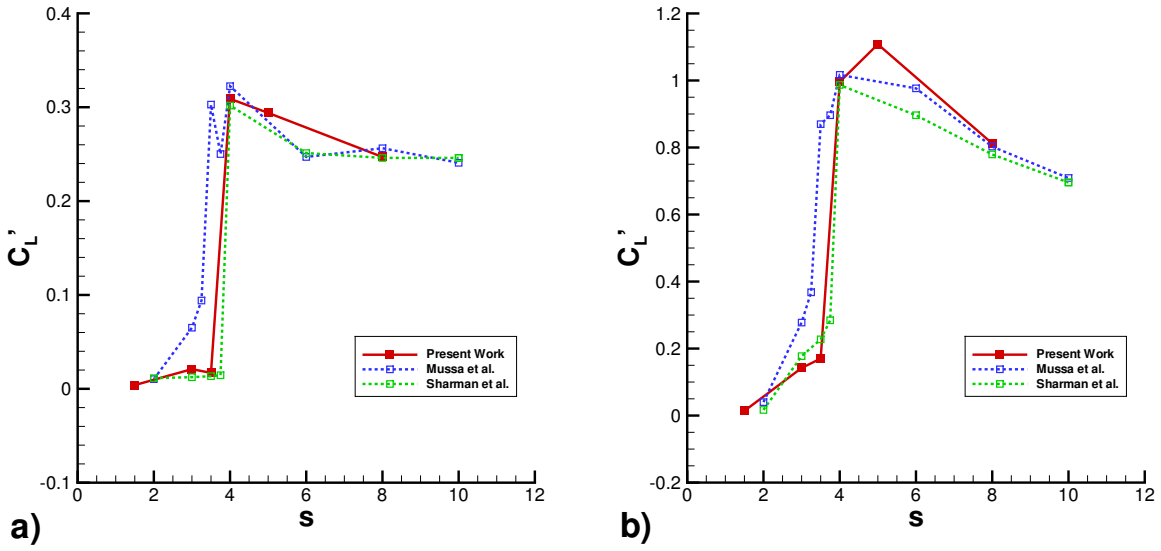

Figure 3.5: C_L' against cylinder spacing, at Re=100. a) upstream cylinder; b) downstream cylinder.

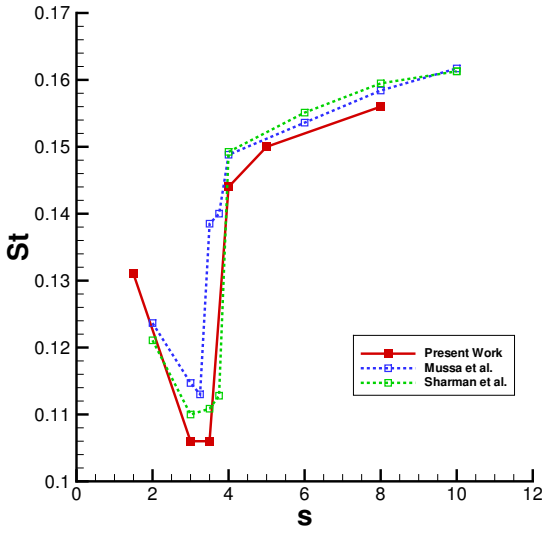

Figure 3.6: Strouhal number against cylinder spacing, at Re=100

Table 3.2: 2D Numerical results for flow over one and two circular cylinders in tandem at $160 \leq Re \leq 220$. $s = 0d$ refers to the single cylinder case.

Re	s/d	$\overline{C}_{D1}$	C'_{D1}	C'_{L1}	$\overline{C}_{D2}$	C'_{D2}	C'_{L2}	St
160	0	1.359	0.019	0.406	-	-	-	0.181
160	1.5	1.128	0.002	0.014	-0.171	0.002	0.032	0.145
160	3.0	1.074	0.003	0.017	-0.102	0.005	0.161	0.122
160	3.5	1.060	0.003	0.031	-0.018	0.010	0.267	0.123
160	4.0	1.264	0.032	0.489	0.444	0.115	1.171	0.159
160	5.0	1.286	0.026	0.420	0.513	0.101	1.095	0.167
160	8.0	1.323	0.022	0.414	0.462	0.082	0.776	0.176
180	0	1.366	0.025	0.452	-	-	-	0.186
180	1.5	1.115	0.002	0.014	-0.188	0.002	0.033	0.153
180	3.0	1.058	0.003	0.021	-0.114	0.007	0.193	0.127
180	3.5	1.040	0.003	0.030	-0.043	0.011	0.269	0.124
180	4.0	1.275	0.037	0.525	0.438	0.122	1.200	0.166
180	5.0	1.291	0.034	0.456	0.491	0.114	1.107	0.172
180	8.0	1.331	0.029	0.455	0.409	0.088	0.783	0.180
200	0	1.376	0.032	0.495	-	-	-	0.190
200	1.5	1.107	0.002	0.016	-0.201	0.001	0.041	0.161
200	3.0	1.043	0.003	0.023	-0.129	0.009	0.212	0.127
200	3.5	1.023	0.003	0.028	-0.068	0.012	0.267	0.123
200	4.0	1.287	0.041	0.558	0.442	0.135	1.229	0.170
200	5.0	1.295	0.041	0.489	0.459	0.125	1.111	0.175
200	8.0	1.340	0.034	0.490	0.365	0.093	0.805	0.184
220	0	1.387	0.039	0.533	-	-	-	0.194
220	1.5	1.100	0.002	0.018	-0.214	0.000	0.045	0.166
220	3.0	1.031	0.004	0.025	-0.144	0.010	0.222	0.128
220	3.5	1.009	0.003	0.028	-0.090	0.014	0.263	0.124
220	4.0	1.296	0.046	0.584	0.432	0.146	1.247	0.173
220	5.0	1.301	0.043	0.521	0.429	0.131	1.112	0.178
220	8.0	1.350	0.040	0.521	0.329	0.103	0.844	0.188

3.3.2 Mean and RMS Force Coefficients and Strouhal Number in 3D

For the 3D simulations, the span-wise dimension was set to $6.0d$, with span-wise periodicity. This dimension was chosen since the stream-wise vortex pairs from the Mode A instabilities are expected to have wavelengths of roughly $3d - 4d$ [70]. The same criterion was used to assess convergence to a statistically stationary state. However, the single cylinder case and $s \geq 4d$ cases at $Re = 220$ showed low frequency oscillations in the signals (periods of ~ 30 shedding cycles), making it difficult to achieve the $\pm 1\%$ constancy (Figure 3.16). In such cases, simulations were extended to record at least 80 shedding cycles. Also note that the force coefficients presented in this paper are span-wise averaged values. The results are summarized in Table 3.3.

The relative differences with the 2D cases can be used as a measure for the 3D effects in the flow. From the single cylinder cases ($s = 0d$), it can be seen that the 3D case at $Re = 220$ displays notable discrepancies with the 2D case. This may be regarded as evidence for 3D flow structures, suggesting that the transition to 3D flow occurs at $200 < Re_c \leq 220$. Figure 3.7 compares the flow fields at $Re = 200$ (2D flow) and $Re = 220$ (3D flow). The figure clearly indicates that the vortices being shed from the cylinder at $Re = 200$ are 2D, while the $Re = 220$ case shows the deformation in the span-wise vortices, which is characteristic of the Mode A instabilities. According to Barkley and Henderson's Floquet stability analysis [5], this transition to 3D flow should occur at $Re_c = 188.5$, which raises the possibility of insufficient grid resolution for the present simulation. However, the trend where the 3D computation for $Re = 220$ yielded a lower St, $\overline{C}_D$, and C'_L are consistent with the findings by [37]. It is also interesting to see the rapid increase in C'_D in the 3D simulation, which is likely due to the low frequency fluctuations in the C_D time history, as will be shown in the following section (Figure 3.17 in Section 3.3.3).

When there are two cylinders in tandem, notable differences were observed at $Re = 220$, mainly for the cases in the WG regime ($s > s_c$) (the exceptional case of $s = 3d$ is discussed in the following paragraph). Figures 3.8 to 3.11 compare the differences between the 2D and 3D computations. Figure 3.8 suggests that for $s > s_c$, the 2D computations under-predict $\overline{C}_D$ for the upstream cylinder, and over-predict for the downstream cylinder. Figure 3.9 shows that the 3D simulations yield lower values of C'_D for $s = 4d$ and $s = 5d$, but jumps to a higher value at $s = 8d$, quickly approaching the prediction for the single cylinder case. It also shows that in the WG regime, the 3D predictions for C'_D are lower than the single cylinder case, which contrasts the 2D predictions that are higher than the single cylinder case. This suggests that the downstream cylinder has a stabilizing effect on C'_D, provided that the span-wise modes are allowed to exist. In terms of C'_L, the 3D simulations

Table 3.3: 3D Numerical results for flow over one and two circular cylinders in tandem at $160 \leq Re \leq 220$. $s = 0d$ refers to the single cylinder case. Numbers in parentheses denote the % difference with respect to the 2D case.

Re	s/d	$\overline{C}_{D1}$		C'_{D1}		C'_{L1}		$\overline{C}_{D2}$		C'_{D2}		C'_{L2}		St	
160	0	1.359	(0.0)	0.019	(0.0)	0.406	(0.0)	-	(-)	-	(-)	-	(-)	0.181	(0.0)
160	1.5	1.128	(0.0)	0.002	(0.0)	0.014	(0.0)	-0.171	(0.0)	0.002	(0.0)	0.032	(0.0)	0.145	(0.0)
160	3	1.074	(0.0)	0.003	(0.0)	0.017	(0.0)	-0.102	(0.0)	0.005	(0.0)	0.161	(0.0)	0.122	(0.0)
160	3.5	1.061	(0.0)	0.003	(0.0)	0.031	(0.0)	-0.018	(0.0)	0.010	(0.0)	0.268	(0.4)	0.123	(0.0)
160	4	1.264	(0.0)	0.032	(0.0)	0.487	(-0.4)	0.442	(-0.5)	0.115	(0.0)	1.173	(0.2)	0.159	(0.0)
160	5	1.286	(0.0)	0.027	(0.4)	0.420	(0.0)	0.514	(0.2)	0.101	(0.0)	1.095	(0.0)	0.167	(0.0)
160	8	1.323	(0.0)	0.022	(0.0)	0.414	(0.0)	0.465	(0.6)	0.083	(1.2)	0.778	(0.3)	0.176	(0.0)
180	0	1.366	(0.0)	0.025	(0.0)	0.452	(0.0)	-	(-)	-	(-)	-	(-)	0.186	(0.0)
180	1.5	1.115	(0.0)	0.002	(0.0)	0.014	(0.0)	-0.188	(0.0)	0.002	(0.0)	0.033	(0.0)	0.153	(0.0)
180	3	1.058	(0.0)	0.003	(0.0)	0.021	(0.0)	-0.114	(0.0)	0.007	(0.0)	0.193	(0.0)	0.126	(-0.8)
180	3.5	1.040	(0.0)	0.003	(0.0)	0.030	(0.0)	-0.043	(0.0)	0.011	(0.0)	0.269	(0.0)	0.124	(0.0)
180	4	1.275	(0.0)	0.037	(0.0)	0.525	(0.0)	0.438	(0.0)	0.121	(-0.8)	1.200	(0.0)	0.166	(0.0)
180	5	1.291	(0.0)	0.034	(0.0)	0.456	(0.0)	0.490	(0.0)	0.114	(0.0)	1.109	(0.1)	0.172	(0.0)
180	8	1.330	(0.0)	0.029	(0.0)	0.454	(-0.2)	0.408	(-0.2)	0.088	(0.0)	0.781	(-0.1	0.181	(0.5)
200	0	1.376	(0.0)	0.032	(0.0)	0.495	(0.0)	-	(-)	-	(-)	-	(-)	0.189	(-0.5)
200	1.5	1.107	(0.0)	0.002	(0.0)	0.016	(0.0)	-0.201	(0.0)	0.001	(0.0)	0.041	(0.0)	0.161	(0.0)
200	3	1.043	(0.0)	0.003	(0.0)	0.023	(0.0)	-0.130	(-0.8)	0.009	(0.0)	0.209	(-1.4)	0.127	(0.0)
200	3.5	1.023	(0.0)	0.003	(0.0)	0.028	(0.0)	-0.068	(0.0)	0.012	(0.0)	0.268	(0.4)	0.123	(0.0)
200	4	1.267	(-1.5)	0.037	(-10.0)	0.532	(-4.7)	0.482	(9.1)	0.109	(-19.5)	1.210	(-1.6)	0.169	(-0.5)
200	5	1.296	(0.0)	0.041	(0.0)	0.490	(0.2)	0.461	(0.4)	0.124	(-0.8)	1.114	(0.3)	0.175	(0.0)
200	8	1.340	(0.0)	0.034	(0.0)	0.489	(-0.2)	0.366	(0.3)	0.093	(0.0)	0.804	(-0.1)	0.184	(0.0)
220	0	1.322	(-4.7)	0.058	(49.7)	0.426	(-20.0)	-	(-)	-	(-)	-	(-)	0.188	(-3.1)
220	1.5	1.100	(0.0)	0.002	(0.0)	0.018	(0.0)	-0.214	(0.0)	0.000	(0.0)	0.045	(0.0)	0.166	(0.0)
220	3	1.033	(0.2)	0.004	(0.0)	0.028	(12.0)	-0.128	(11.1)	0.011	(10.0)	0.246	(10.8)	0.128	(0.0)
220	3.5	1.009	(0.0)	0.003	(0.0)	0.028	(0.0)	-0.091	(-1.1)	0.014	(0.0)	0.264	(0.4)	0.123	(-0.8)
220	4	1.275	(-1.6)	0.041	(-10.9)	0.554	(-5.1)	0.477	(10.4)	0.119	(-18.5)	1.214	(-2.6)	0.169	(-2.3)
220	5	1.275	(-2.0)	0.040	(-7.0)	0.479	(-8.1)	0.479	(11.7)	0.117	(-10.7)	1.066	(-4.1)	0.173	(-2.8)
220	8	1.308	(-3.1)	0.055	(37.5)	0.457	(-12.3)	0.434	(31.9)	0.144	(39.8)	0.798	(-5.5)	0.182	(-3.2)

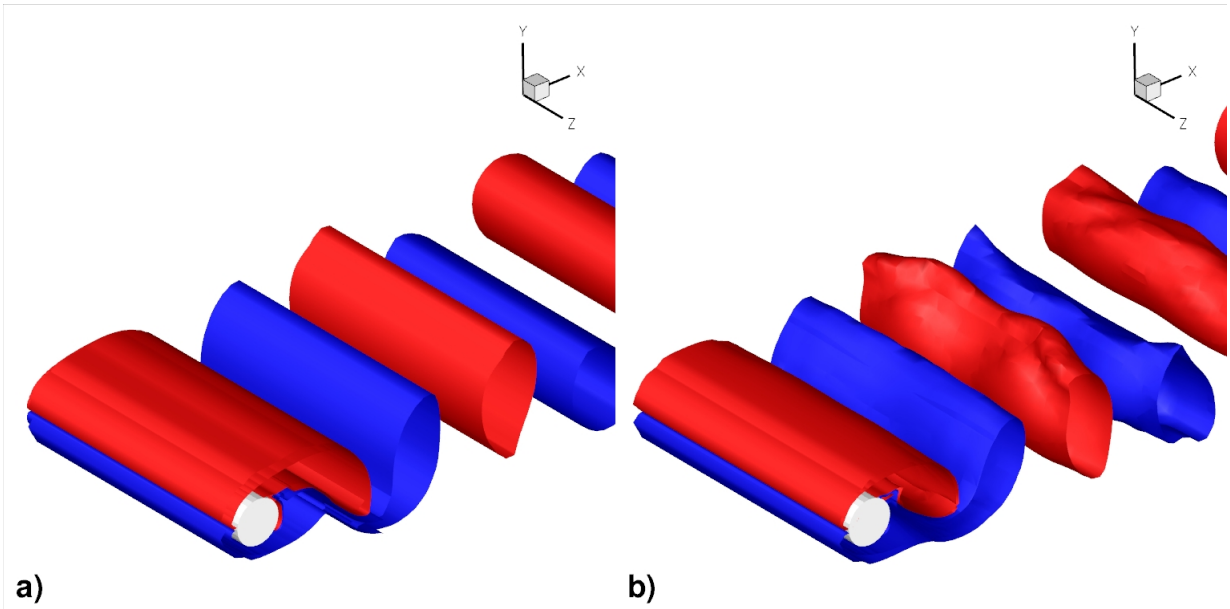

Figure 3.7: Isosurface contours of span-wise vorticity (ω_z) for single cylinder case. Red: $\omega_z = 0.5$; blue: $\omega_z = -0.5$. a) $Re = 200$; b) $Re = 220$;

show lower values for both cylinders for $s > s_c$ (Figure 3.10), which is similar to what was observed for the single cylinder case. From Figure 3.11, it can be seen that St is consistently lower in the 3D case for $s > s_c$. Figure 3.11 also compares the present results with those from Deng *et al.* [13]. Although the 2D results from the present study show significantly lower values for the WG regime, there is good agreement in the general trends and the values obtained from the 3D simulation. In general, the 3D effects were suppressed for small s, and more pronounced for $s > s_c$. An important observation that was made here is that for the upstream cylinder, the differences in the force coefficients between the 2D and 3D simulations were always smaller than the differences observed in the single cylinder case. This suggests that although the presence of the downstream cylinder promotes the inception of the Mode A instabilities, the 3D effects on the force coefficients of the upstream cylinder are smaller than that of a single cylinder flow.

There were some disagreements between the 2D and 3D simulations for the $s = 4d$ case at $Re = 200$ and the $s = 3d$ case at $Re = 220$ as well. For the $s = 4d$ case at $Re = 200$, the presence of 3D flow was confirmed from its instantaneous flow field (Figure 3.12). This is likely to be due to s being large enough to form the WG regime, but small enough that the two cylinders do not act as isolated cylinders. The proximity of the cylinders increases

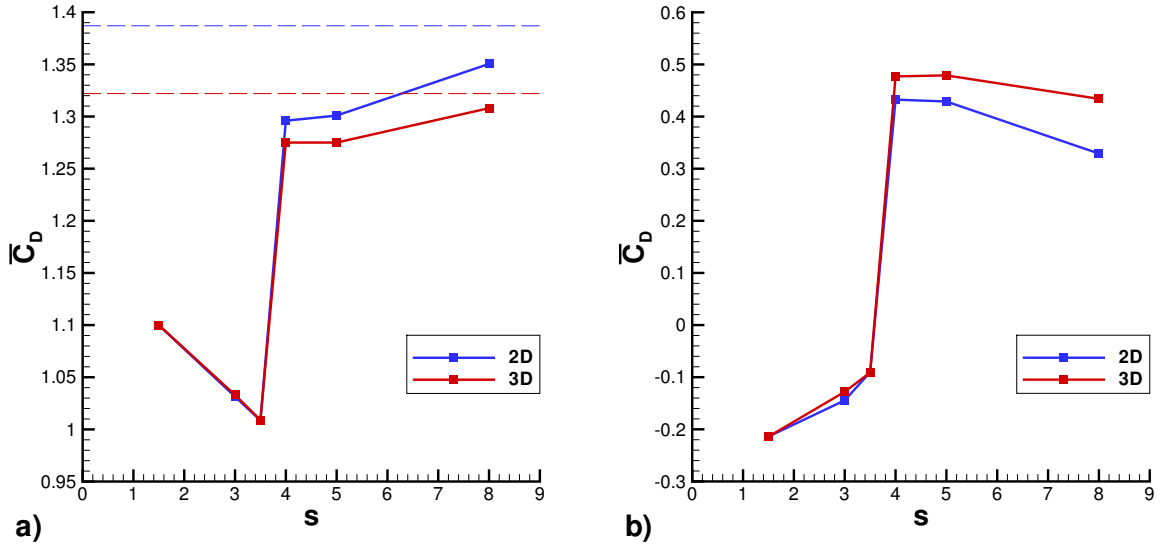

Figure 3.8: $\overline{C}_D$ against s at $Re = 220$ for 2D and 3D computations. Horizontal dashed line in a) indicate the 2D and 3D values for a single cylinder case at $Re = 220$. a) upstream cylinder; b) downstream cylinder.

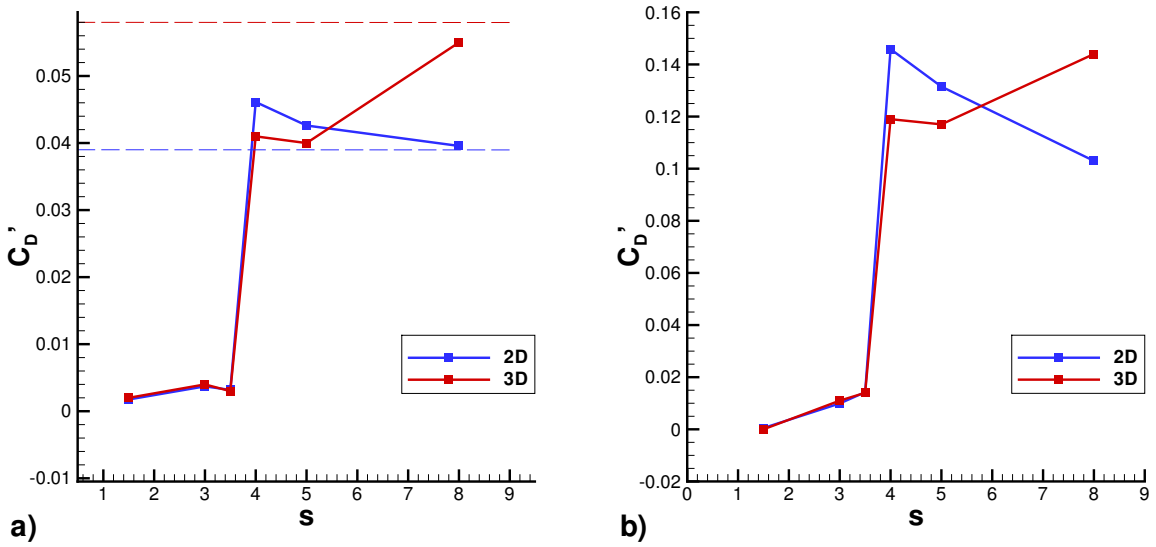

Figure 3.9: C'_D against s at $Re = 220$ for 2D and 3D computations. Horizontal dashed line in a) indicate the 2D and 3D values for a single cylinder case at $Re = 220$. a) upstream cylinder; b) downstream cylinder.

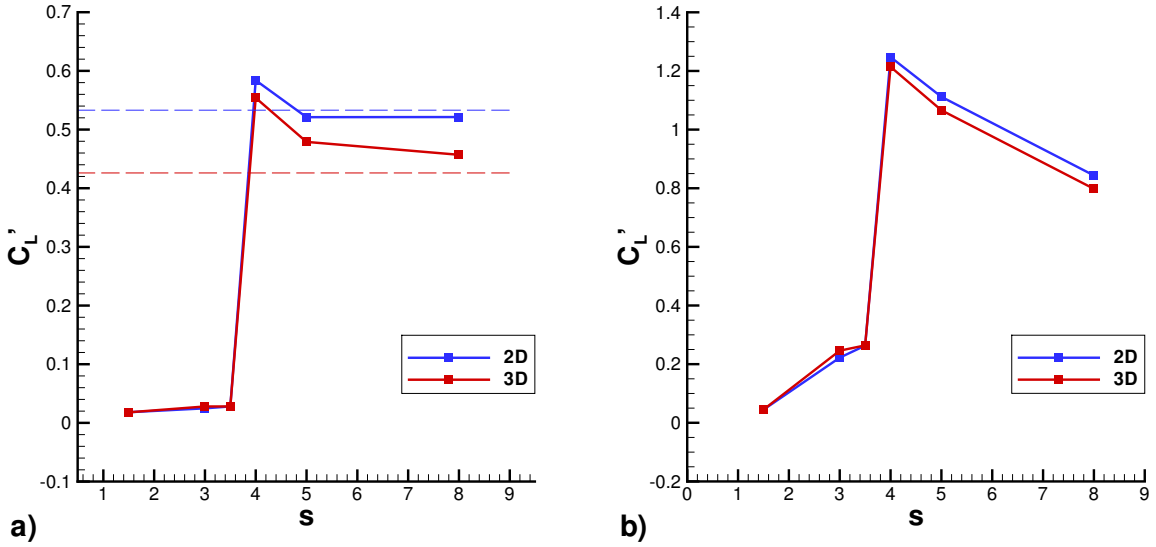

Figure 3.10: C'_L against s at $Re = 220$ for 2D and 3D computations. Horizontal dashed line in a) indicate the 2D and 3D values for a single cylinder case at $Re = 220$. a) upstream cylinder; b) downstream cylinder.

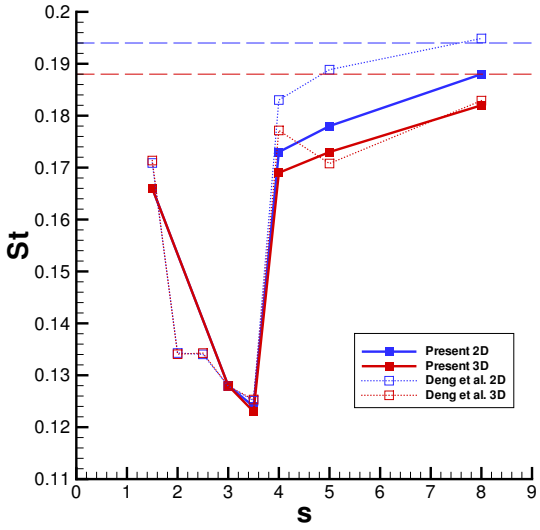

Figure 3.11: St against s at $Re = 220$ for 2D and 3D computations. Horizontal dashed line indicates the 2D and 3D values for a single cylinder case at $Re = 220$. Comparison with results from Deng *et al.* [13].

the influence of the adverse pressure gradient in the interstitial region on the near wake of the upstream cylinder, thus triggering the Mode A instabilities at a lower Re than the isolated cylinder case [10]. For the $s = 3d$ case at $Re = 220$, the flow displayed a slightly different 3D flow structure than those associated with the Mode A instabilities (Figure 3.13). The span-wise wavelength of the unstable mode is $\sim 6d$, and since the flow is in the AG regime, it is likely to be the T3 mode identified by Carmo *et al.* [10]. The fact that this instability was observed at $s = 3d$ and not at $s = 3.5d$, which is also in the AG regime, is also consistent with the Re_c trends reported in [10]. It was found that the T3 mode increases C'_L for both cylinders, which is contrary to the trends observed with the cases that displayed the Mode A instabilities, but no significant change was observed in St.

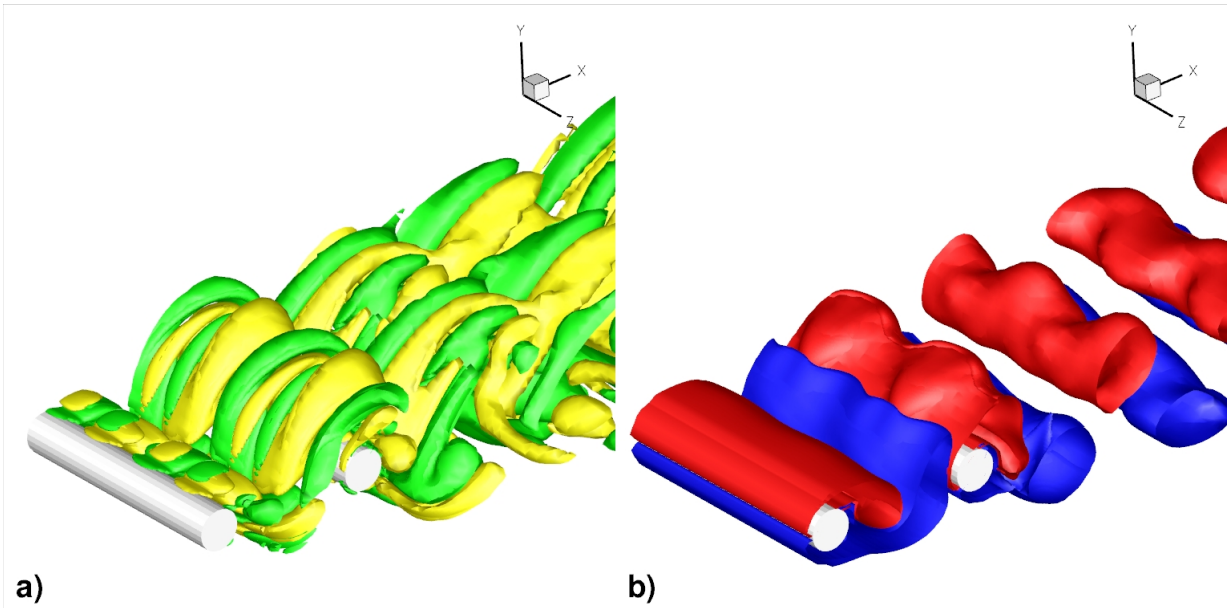

Figure 3.12: Isosurface contours of: a) stream-wise vorticity (yellow: $\omega_x = 0.1$, green: $\omega_x = -0.1$), b) span-wise vorticity (red: ω_z=0.5, blue: $\omega_z = -0.5$), for $s = 4d$ cylinder case at $Re = 200$. Span-wise dimension in the figure is $6.0d$.

3.3.3 Time Histories of Force Coefficients

In the previous section, it was found that the 3D effects can increase or decrease C'_D for both the upstream and downstream cylinders, depending on the cylinder spacing. When

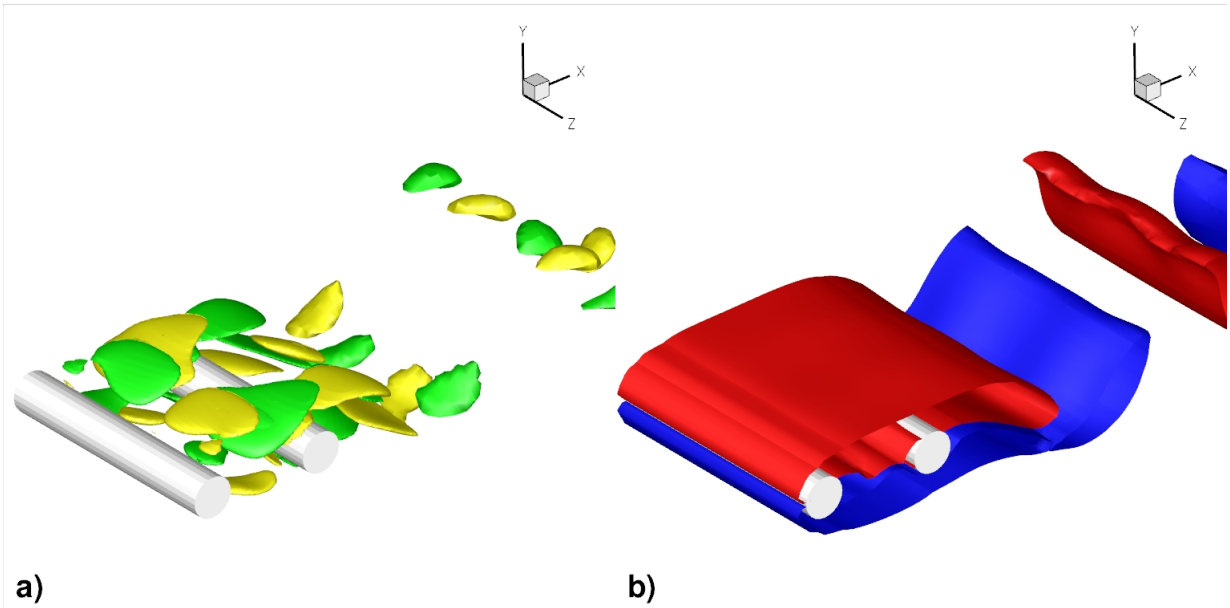

Figure 3.13: Isosurface contours of: a) stream-wise vorticity (yellow: $\omega_x = 0.1$, green: $\omega_x = -0.1$), b) span-wise vorticity (red: ω_z=0.5, blue: $\omega_z = -0.5$), for $s = 3d$ cylinder case at $Re = 220$. Span-wise dimension in the figure is $6.0d$.

the cylinders are $4d \leq s \leq 5d$, the introduction of the span-wise instabilities decreases C'_D, but when the spacing is increased to $s = 8d$, the values jump to a much higher value than the 2D prediction. In order to explicate this phenomenon, the time histories of the force coefficients at $Re = 220$ with s of $4d$, $5d$, and $8d$, as well as the single cylinder case were compared (Figures 3.14–3.17). Note that for $s = 5d$, $s = 8d$, and the single cylinder case, the time histories only show 160 time units, while the 3D simulations show 480 time units. This is because the 3D simulations were run for a longer duration than the 2D simulations to accommodate for the unsteady fluctuations in the flow.

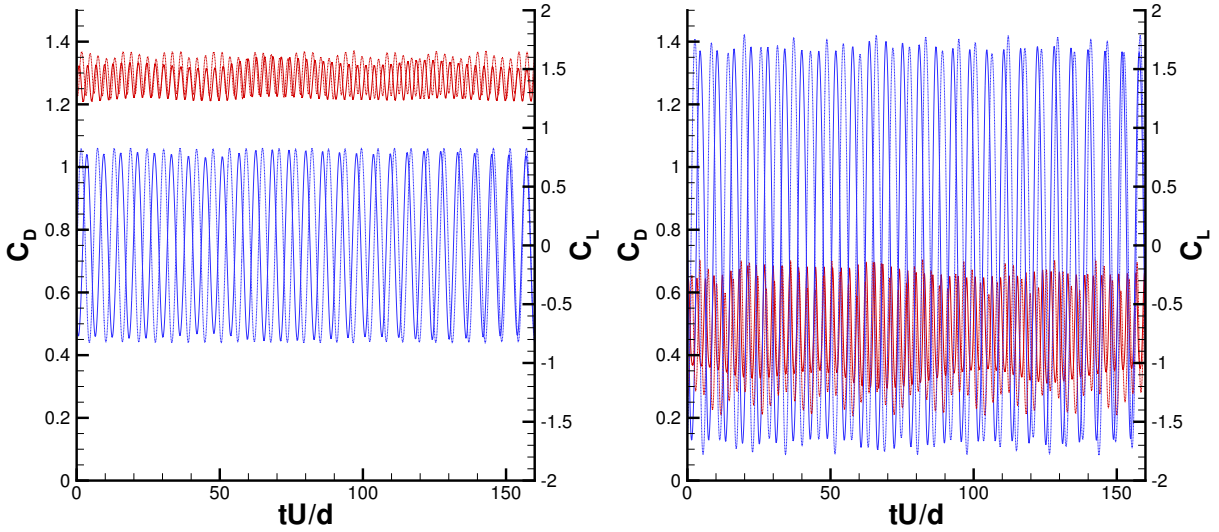

Figure 3.14: Time histories of the force coefficients in 2D and 3D at $Re = 220$ with $s = 4d$. Red: C_D; blue: C_L. $\cdots$: 2D; —: 3D. a) upstream cylinder; b) downstream cylinder.

From Figures 3.16 and 3.17, it can be observed that the upstream cylinder for $s = 8d$ shows very similar trends with the single cylinder case, suggesting that with large spacings, the influence of the downstream cylinder diminishes, and the upstream cylinder acts as a single isolated cylinder. For these cases, the 3D effects in the flow introduce temporal variation in the amplitudes of fluctuating forces from the primary vortex shedding, and superimposes an additional fluctuating component (repetition periods of $\sim$ 250 time units) onto C_D. Since C_L in the 3D flow contains time periods with lower amplitudes, the effective C'_L is reduced. C_D also shows some reduction in the amplitudes at the Strouhal frequency, but the low frequency pulsations in the local mean C_D contributes to an increased C'_D. When

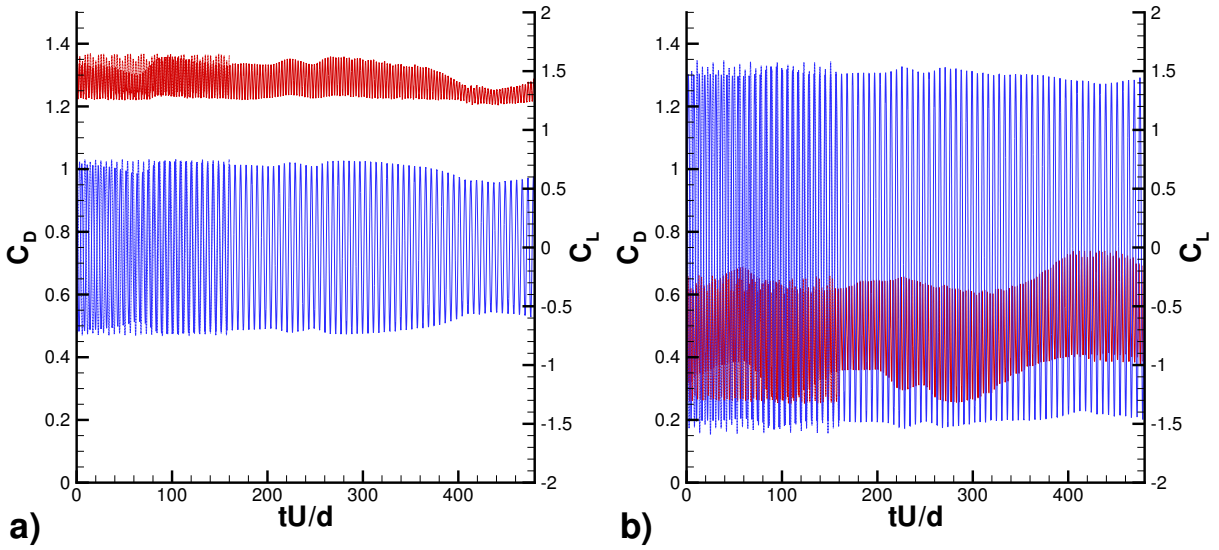

Figure 3.15: Time histories of the force coefficients in 2D and 3D at $Re = 220$ with $s = 5d$. Red: C_D; blue: C_L. $\cdots$: 2D; —: 3D. a) upstream cylinder; b) downstream cylinder.

$s = 5d$ (Figure 3.15), the unsteady pulsations have a much lesser effect than at $s = 8d$, and the Strouhal fluctuations are smaller than the 2D case, leading to an overall reduction in C'_D. At $s = 4d$ (Figure 3.14), the effects of the unsteady oscillations are almost negligible, while there is a notable reduction in the amplitudes of the Strouhal fluctuations, hence yielding a considerably lower C'_D than its corresponding 2D case.

3.4 Summary

In this study, the LBM was used to simulate the flow over one and two cylinders in tandem, to analyze the effects of the early 3D instabilities on the force coefficients and St. For a single cylinder, the flow transitioned to 3D between $200 < Re \leq 220$, while for the tandem cylinders, the $s = 4d$ configuration displayed 3D flow at $Re = 200$. At $Re = 220$, the flow was 3D for $s > s_c$, and also for $s = 3d$, where the presence of the T3 mode was confirmed. Although the precise Re for transition to 3D flow may deserve more investigation, the relative order at which transition occurs for different cylinder spacings was in good agreement with previous studies. By comparing the force coefficients obtained

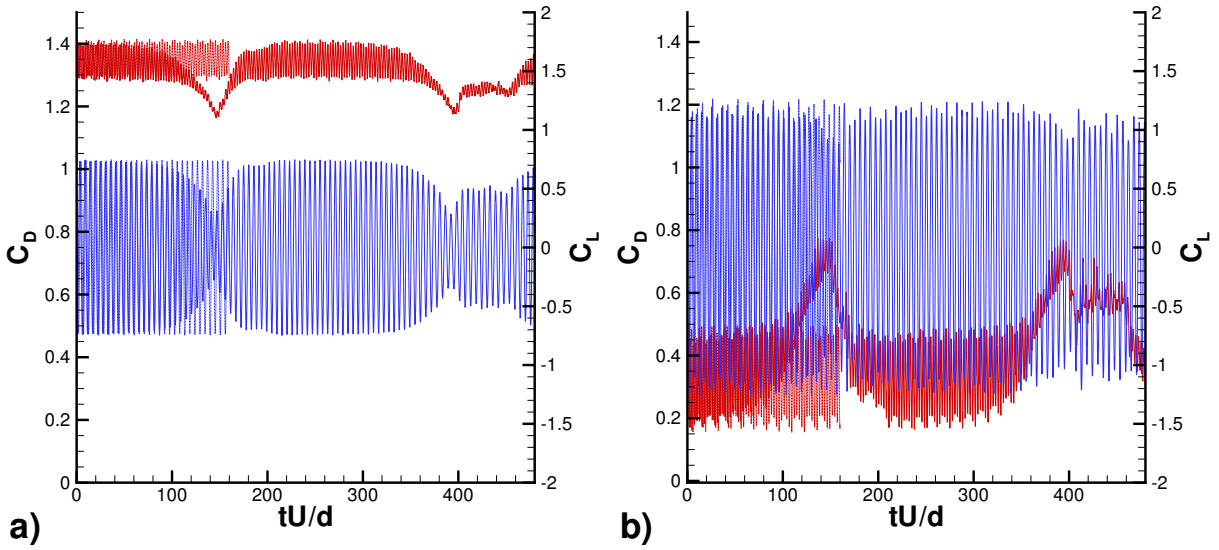

Figure 3.16: Time histories of the force coefficients in 2D and 3D at $Re = 220$ with $s = 8d$. Red: C_D; blue: C_L. $\cdots$: 2D; —: 3D. a) upstream cylinder; b) downstream cylinder.

from the 2D and 3D cases, it was found that for the Re studied, the presence of the downstream cylinder promotes the inception of the Mode A instabilities, but the 3D effects on the force coefficients of the upstream cylinder are smaller than that of a single cylinder flow. It was also found that the T3 modes affect the force coefficients in a different way than the Mode A instabilities.

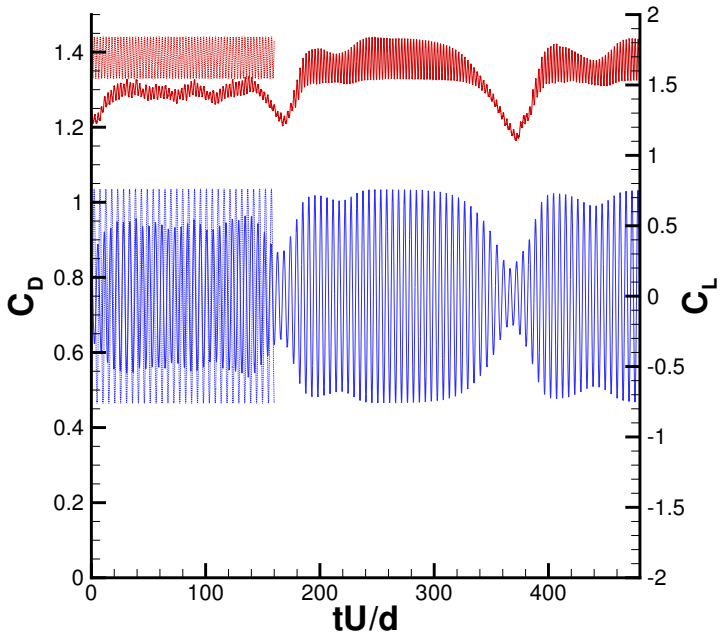

Figure 3.17: Time histories of the force coefficients in 2D and 3D at $Re = 220$ for one cylinder. Red: C_D; blue: C_L. $\cdots$: 2D; —: 3D.

Chapter 4

The Lattice Boltzmann Method on the GPU

In order to perform reliable simulations involving turbulence, it is necessary to have sufficient grid resolution in the simulation domain, which leads to an increased computational load. However, such penalties can be alleviated by the use of innovative hardware, such as the GPU. Although GPUs have many restrictions in its operation and programming, there has been an increased interest in using them for general purpose calculations, for its high processing power that it harnesses from its massively multi-core architecture. NVIDIA's Tesla M2070 for example, is rated with a peak performance of 1331 GFLOPs with a peak memory bandwidth of 177 GB/s (with ECC off) [47].

Since one of the main advantages of the LBM is the data locality in the algorithm, there has been an appreciable amount of effort put into realizing efficient LBM implementations on the GPU [3, 6, 16, 23, 24, 42, 49, 54, 64, 65]. In this chapter, the implementation of the LBM on the GPU using NVIDIA's Compute Unified Device Architecture (CUDA), a programming architecture developed specifically to utilize the processing power of GPUs, is described.

4.1 CUDA

Contrary to CPUs, GPUs are designed specially for highly parallel computation, housing several *Streaming Multiprocessors* (SMs). Each SM consists of a set of *Scalar Processors*

(SPs), an instruction unit, and shared memory. In the Fermi architecture (compute capability 2.x), it also includes an L1 cache. The following section discusses the CUDA programming model. All information presented here is based on the NVIDIA CUDA C Programming Guide [46] and NVIDIA CUDA C Best Practices Guide [45].

4.1.1 Process Hierarchy

In order to allow GPU programs to scale efficiently across GPUs with different numbers of processing units, the CUDA programming model employs a hierarchy of process threads. The smallest process unit is called the *thread*, and GPU functions (referred to as *kernels*), are executed by each thread concurrently. Threads are grouped into *blocks*, which is a process unit used to allocate jobs to the available SMs. The threads in a block can synchronize and communicate through shared memory, as they will always be executed within the same SM. Blocks are organized in a *grid*, which is a process unit used to execute kernels on a group of threads. For devices with compute capabilities 2.0 and above, both blocks and grids can have up to three dimensions, making it easy to decompose a 3D computational domain into spatially organized blocks.

Each SM executes a kernel when they are assigned a thread block. Note that several blocks can be assigned to a single SM, if there are enough resources. The SM first decomposes the thread block into thread groups of 32, which are referred to as *warps*. A SM can only process a single warp at a given instance, but it can switch to a different warp if the current warp becomes idle (due to a memory request etc.). Having several warps to switch to can help hide latencies from memory transactions, so one optimization strategy would be to maximize the number of warps that reside in a SM, thus increasing the *occupancy*. This is equivalent to maximizing the number of threads that reside in the SM, and can be achieved by adjusting the register and shared memory usage for each block as well as the number of threads assigned per block.

4.1.2 Memory Hierarchy

In GPU computing, the different types of memory are also organized in a hierarchical fashion. Each thread has its own private local memory, accessible only by the thread itself. Within a thread block, threads have common access to shared memory, which has the same lifetime as the block. This means that once a block finishes its execution, the contents of the shared memory are destroyed. Global memory, or device memory, is the largest memory space, and is accessible by the CPU as well as all threads on the GPU, and its scope spans

the lifetime of the application. Global memory also includes the constant memory and texture memory, which are both read-only. For devices with compute capabilities 2.0 and above, there also exists an L1 cache for each SM, as well as an L2 cache that is shared by all SMs.

4.1.3 Basic Optimizations

One of the most important considerations when programming for GPU computing is the memory management. The memory transactions involving registers and shared memory are fast, and normally do not have significant impact on overall performance. Global memory transactions on the other hand, have latencies of several hundred clock cycles, and are sensitive to memory access patterns. Hence, optimizing global memory transactions is critical in improving the performance of a program running on a GPU. This is especially the case for memory bound algorithms such as the LBM.

Global memory accesses are 32, 64, or 128 byte memory transactions of 1, 2, 4, 8, or 16 byte words, issued by half-warps (compute capability 1.x) or warps (compute capability 2.x). If the memory requests from a half-warp (or warp) all reside in a continuous memory location where its first address is a multiple of the transaction size, the 16 memory accesses can be combined into a single transaction. Memory transactions that fulfill these requirements are called *coalesced* accesses, and are the optimal form for accessing global memory. For example, if the threads from a half-warp (indexed as n=0,...,15) each need access to 4 byte floats that are stored in global memory with addresses of Base Address + 4 bytes*n, and Base Address is a multiple of 4 bytes*16, the memory access is fully coalesced. This example will result in a single memory transaction of 64 bytes. The consequences from uncoalesced transactions are dependent on the compute capability of the device. For earlier devices with compute capabilities of 1.0 or 1.1, all transactions that do not meet the sizing and base address requirements result in 16 separate transactions. For compute capabilities 1.2 and above, the penalties are less severe. Memory transactions that are contained in the segment size will be combined into one transaction, regardless of whether the base address is a multiple of the segment size, and the transaction sizes will be adjusted to minimize over-fetching. For compute capabilities 2.0 and above, the global memory fetches are cached to L1 and L2 caches. The L1 cache operates with a 128 byte cache line that maps to a 128 byte aligned segment in global memory, while the L2 cache has a smaller transaction size at 32 bytes. For misaligned accesses, the L1 cache can improve the transaction efficiency, but for scattered accesses, disabling the L1 cache to reduce data over-fetch can prove to be faster.

Therefore, in terms of memory access patterns, it is important to organize the data layout and memory fetching patterns in a way that promotes coalesced accesses. The high latencies on the other hand can be hidden by increasing the occupancy of the SMs. This means the SMs should have multiple active warps, so that the processors can execute instructions on a different warp while the current warp is idle. More details on the programming model and common optimization strategies can be found in the NVIDIA CUDA C Programming Guide [46] and NVIDIA CUDA C Best Practices Guide [45].

4.2 The lattice Boltzmann method in CUDA

The LBM is known to be a suitable method for implementing on GPUs due to its data locality. In a typical implementation, each lattice node in the domain is assigned a thread, and kernels are invoked to perform the propagation and collision steps. These steps should be fused into one process, as it will reduce the number of global memory accesses by a factor of two. The main challenge in the GPU implementation of the LBM is the memory transactions associated with the propagation step. During propagation, threads must access the distribution functions stored in its neighboring nodes. In order to allow coalesced memory accesses during this step, it is important to arrange the data in a Structure of Arrays (SoA), as opposed to an Array of Structures (AoS). In addition, by aligning the array size to a multiple of the half-warp size, one can ensure optimal memory accesses for propagation in the major direction. However, propagation in the minor direction causes misaligned memory transactions that can severely deteriorate performance on some devices. The incorporation of local grid refinements further adds complexity to the implementation, mainly due to the interface handling that involves additional memory transactions between nodes that are not necessarily aligned.

4.2.1 Uniform Grid Implementations

This section first outlines some of the recent implementations the LBM on uniform grids. The performance of each method is given in terms of Million Lattice Updates Per Second (MLUPS), which is a common performance measurement used for the LBM. Tölke and Krafczyk [64, 65] solve the memory misalignment problem by splitting the domain into an array of one dimensional blocks, and performing the propagations that involve shifts in the minor directions through shared memory. Since the scope of shared memory is limited to the block, the distributions that are leaving the block along the minor direction are temporarily stored as an incoming distribution from the opposite side of the block. A

separate kernel with a different block topology is then invoked to place these distributions to their correct locations. This approach ensures that all global memory accesses are fully coalesced, but has the disadvantages of requiring an additional kernel to be launched. Their implementation achieved a maximum performance of 592 MLUPS for the D3Q13 lattice on the GeForce 8800 Ultra.

Obrecht et al. [49] showed that a relatively straight forward "reversed" scheme can achieve high memory bandwidths on devices with higher compute capabilities. This method relies on the fact that the penalties incurred from misaligned memory accesses have been alleviated on the higher compute capability devices (1.2 and above), and on the observation that misaligned writes are more costly than misaligned reads. The implementation consists of each thread performing a series of global memory reads that involve misalignments, applying the collision operator, and writing the updated distributions onto global memory in a fully coalesced fashion. Their method showed a maximum performance of 516 MLUPS for the D3Q19 lattice on the GeForce GTX 295.

Both of the implementations above allocate arrays for two sets of domains, and alternate between the two grids after every time step to avoid data overwriting. There have also been work that performs the reads and writes in place, allowing the simulation to be contained in a single array, thereby reducing memory requirements by half. Myre et al. [42] compared the "ping-pong" scheme that alternates between two sets of grids, with the "flip-flop" and "Lagrangian" schemes that only use one set of data. The "ping-pong" pattern used in their study is similar to the "reversed" scheme used by Obrecht et al., but it performs the collision in place first, and then writes the updated distributions to the neighboring sites to complete the propagation. The "Lagrangian" pattern assigns a global memory location for each fluid packet, and the threads are assigned a fixed point in the simulation domain. At each time step, the threads selectively read the fluid packets that would arrive at its location, perform the collision step, and write the updated values to the original memory locations. In the "flip-flop" pattern, two different read/write methods are used alternately for each time step. In the first time step, each thread reads the distribution functions from its node, applies the collision operator, and writes the updated information in the same node location, but with reversed directions. In the second time step, the threads now read the incoming distributions from the adjacent nodes, and subsequently writes to the adjacent nodes. Their study showed that the "Lagrangian" method is the fastest among the three memory access patterns, which achieved a maximum performance of 444 MLUPS for the D3Q19 lattice on the Tesla C1060.

Astorino et al. [3] also present a single grid method that employs the swapping technique [33] for propagation. This technique involves a series of swapping operations between distributions in the opposite directions on adjacent nodes, thus eliminating data overwriting.

The limitation with this propagation method in the context of GPU computing is that it does not allow the collision and propagation steps to be combined into one kernel. This is because the swapping technique assumes knowledge of the order of the nodes at which the algorithm is applied. Since all threads execute concurrently, one cannot know the order that the nodes are executed *a priori*, and will have to rely on kernel synchronizations to ensure that the collision operation is only applied to the nodes that have completed the swapping with all adjacent nodes. With this method, they report a maximum performance of 375 MLUPS on the D3Q19 lattice using the GeForce GTX 480.

The current implementation of the LBM was based on the "reversed" scheme, where each thread reads incoming distributions from its neighboring nodes, completes the collision step, and writes the updated distributions in a fully coalesced manner [49]. One dimensional thread blocks oriented in a way that it is aligned to the minor dimension of the array (in this case, the x direction) to allow coalesced accesses. The optimal block size is dependent on the size and shape of the domain, but in general, having 64 or 128 threads per block leads to good performance. Below is an excerpt that outlines the procedure for the current LBM implementation.

```
/*
Stripped down code for running the LBM on the GPU using CUDA.
*/
#include <cuda.h>
...
/*
Device function for converting x,y,z coordinates to memory address
Inputs are: f_num = distribution number (0-18); pitch = pitch size of array
*/
inline __device__ int f_mem(int f_num, int x, int y, int z, size_t pitch)
{
        return (x+y*pitch+z*YDIM*pitch)+f_num*pitch*YDIM*ZDIM;
}
...
/*
Kernel for streaming and colliding. Reads distributions from f_in, and
    writes it onto f_out.
Other inputs are: omega = relaxation rate; pitch = pitch size of array
*/
__global__ void update(float* f_in, float* f_out, float omega, size_t pitch)
{
    int x = threadIdx.x+blockIdx.x*blockDim.x;
    int y = threadIdx.y+blockIdx.y*blockDim.y;
    int z = threadIdx.z+blockIdx.z*blockDim.z;
    int j = x+y*pitch+z*YDIM*pitch;
```

```
    float f[19];
    //read from input array. array index computed from device function:
       f_mem
    f[ 0] = f_in[f_mem(0 ,x  ,y  ,z  ,pitch)];
    f[ 1] = f_in[f_mem(1 ,x-1,y  ,z  ,pitch)];
    ...
    f[18] = f_in[f_mem(18,x  ,y+1,z+1,pitch)];
    //apply boundary conditions
    ...
    //apply collision operator
    ...
    //write updated distributions to f_out
    f_out[f_mem(0 ,x,y,z,pitch)] = f[ 0];
    ...
    f_out[f_mem(18,x,y,z,pitch)] = f[18];
}
...
int main()
{
...
//allocate two sets of device memory
float *f_d[2];
//allocate memory
cudaMalloc((void **) &f_d[0], pitch*YDIM*ZDIM*19*sizeof(float));
cudaMalloc((void **) &f_d[1], pitch*YDIM*ZDIM*19*sizeof(float));
//allocate host memory
float *f_h;
//initialize host memory
...
//copy host memory to device memory to initialize simulation
cudaMemcpy2D(f_d[0],pitch,f_h,XDIM*sizeof(float),XDIM*sizeof(float),YDIM*
   ZDIM*19,cudaMemcpyHostToDevice);
cudaMemcpy2D(f_d[1],pitch,f_h,XDIM*sizeof(float),XDIM*sizeof(float),YDIM*
   ZDIM*19,cudaMemcpyHostToDevice);
...
//define thread and block dimensions for GPU
dim3 threads(BLOCKSIZEX, BLOCKSIZEY, BLOCKSIZEZ);
dim3 grid  (XDIM/BLOCKSIZEX,YDIM/BLOCKSIZEY,ZDIM/BLOCKSIZEZ);
...
//define variables to allow pointer switching
int A = 0; int B = 1;
//time loop
for(int t = 0; t<TMAX; t++)
{
    //stream and collide to march in time
```

```
    update<<<grid,threads>>>(f_d[B],f_d[A],omega,pitch);
    //syncrhonize device
    cudaDeviceSynchronize();
    //swap input and output arrays
    swap(A,B);
}
//copy device memory back to host memory
cudaMemcpy2D(f_h,XDIM*sizeof(float),f_d[A],pitch,XDIM*sizeof(float),YDIM*
    ZDIM*19,cudaMemcpyDeviceToHost);
...
//write results to output file
...
//free device memory
cudaFree(f_d[0]);
cudaFree(f_d[1]);
//free host memory
...
return 0;
}
```

4.2.2 Non-uniform Grid Implementations

Schönherr et al. [57] were the first to report on the implementation of a non-uniform grid LBM on the GPU. Their grid refinement scheme is based on the method by Filippova and Hänel [17], and their GPU implementation is based on the shared memory method proposed by Tölke and Krafczyk [64, 65]. For the interpolations involved at the grid interface, they use a compact second order spatial interpolation, where the moments are interpolated from the four (eight for 3D) closest nodes by using a quadratic bubble function, and subsequently converted back into distribution functions [20, 66]. They also avoid using time interpolations by selectively discarding the invalid outer nodes after each fine mesh time step. The details regarding the algorithm used to complete the data transactions for the interpolations were not given, but it involves using a pre-computed list of interpolation cells that provide the indices of the associated nodes at the grid interface.

In this study, grid refinements were achieved by a method similar to that used by Schönherr et al. [57]. Quadratic bubble functions were used to realize second order interpolations in space, and the outer nodes of the refinement patch were discarded appropriately. The challenge associated with implementing grid refinements on the GPU is the increased number of memory accesses, due to each grid node requiring 4 memory reads in 2D, or 8 memory reads in 3D, before performing interpolations. Furthermore, a naive implementation where

each thread accesses the required memory locations to load to its local memory would result in non-coalesced accesses, which would significantly deteriorate the overall performance of the code. Here, the interpolation step will be considered for the coarse to fine mesh interpolation and the fine to coarse mesh interpolation separately.

For the coarse to fine mesh interpolations, it is possible to take advantage of the shared memory space. As mentioned earlier in Section 4.1.2, shared memory is a memory space that can be accessed by all threads within a block. Since there will be an overlap in the coarse node accesses for adjacent fine mesh nodes, one can pre-load all the coarse node information onto shared memory, and then have each thread access the shared memory to perform interpolations. The procedure used in achieving this is outlined as follows.

First, the threads are each assigned to the fine mesh nodes, which are organized in blocks of $(an \times bn \times cn)$, where n is the refinement factor $(\delta x_{coarse}/\delta x_{fine})$, and a, b, c are arbitrary integers. For example, for $n = 2$ and $\{a, b, c\} = \{4, 2, 1\}$, the thread block will map to the nodes indicated by the blue diamond, in Figure 4.1 (for clarity, the figure only shows a 2D section). The relevant coarse mesh nodes that are required for interpolation are shown by the red circles. Since there are more threads than there are coarse mesh nodes that require reading, only the first $(a + 1)(b + 1)(c + 1) = 30$ threads are assigned to read from the coarse grid. Since each thread will be reading from adjacent memory locations (except for at the edges), the memory accesses are mostly coalesced. The coarse grid data are stored in shared memory, and each thread is then used to compute the interpolated values, and write to its associated fine mesh nodes. For the fine to coarse mesh interpolations, the above methodology is of little use, since the adjacent coarse grid nodes will never share any fine mesh nodes for interpolation. Hence, the fine to coarse mesh interpolations were realized by assigning a thread to each coarse node, and simply having the threads access the eight fine mesh nodes individually. This leads to uncoalesced memory accesses, but it is hoped that the L1 cache will help in alleviating some of the performance penalties.

4.2.3 Multi-GPU Implementation

One of the limitations in using GPUs for computation is that the available memory is limited. For the Tesla M2070, the device memory is quoted as 6GB, but it must be noted that the ECC memory protection feature occupies 12.5% of the GPU memory, leaving approximately 5.25GB available for use. In the case with LBM, assuming a two-array method, each node in the simulation domain will require $2 \times 19 \times n$ words of memory, where n is the total number of nodes. For a single precision calculation using floats, which

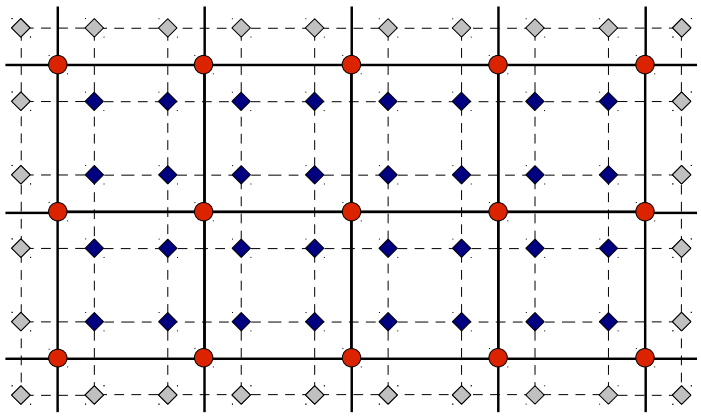

Figure 4.1: Coarse to fine mesh interpolations on the GPU. Thread block maps to the blue diamonds, and interpolation data are read from the red circles.

have a size of 4 bytes, n will be limited to roughly 34×10^6. One way of solving this problem would be to split the domain into multiple zones, and use a separate GPU for each zone. After each time step, the interfacial data can be exchanged via the PCIExpress bus, much like how it is done in MPI parallelizations. The use of multiple GPUs will also lead to a potential improvement in execution speed.

One of the recent studies that employed multiple GPUs for the LBM include Wang and Aoki's large scale multi-node GPU implementation [68]. In their study, the LBM was run using up to 100 Tesla S1070 GPUs, with 2GPUs per node. The inter-GPU communication was managed using a combination of CUDA API `cudaMemcpy` and MPI. They compared the efficiencies of 1D, 2D, and 3D domain partitioning methods, and concluded that when the number of GPUs is <10, 1D decomposition proves to be sufficient, but for >10 GPUs, 2D and 3D partitioning becomes important, as the size of the inter-GPU communication can be reduced. Obrecht et al. [48] also investigated the scalability of multi-GPU LBM codes. Their hardware configuration consisted of a single node connected to eight GPUs. With 1D domain partitioning, they observed a nearly perfect strong scalability for a sufficiently large domain.

In this study, the number of GPUs to be used was limited to two, due to hardware availability. Since the nodes on SHARCNET's 'monk' cluster are each equipped with two Tesla M2070 GPUs, the current implementation does not require the use of MPI for inter-node communication. For inter-GPU communication between GPUs that share the same node, memory transfer is achieved by first copying the source GPU data back to the host memory via `cudaMemcpy`, and subsequently invoking an additional `cudaMemcpy` from host memory

to the destination GPU. This two step procedure can be completed in a single CUDA command by using CUDA's peer-to-peer memory access feature, `cuMemcpyPeerAsync`, which is one of NVIDIA's recent developments to promote multi-GPU programming.

For efficient parallelization, it is critical to perform the inter-GPU memory transfers in an asynchronous fashion to ensure that the communication is overlapped with computation. To do so, two CUDA *streams* are created to separately manage each zone's halo nodes and inner nodes. While the halo data is being exchanged, the inner nodes are marched in time. As long as the inner node computation takes a longer time to complete than the data exchange in the halo region, the computational resources of the GPU will always be fully utilized, thus leading to efficient parallelization. The excerpt below outlines the computational procedure that was taken in this study.

```
...
int main()
{
...
//count number of available GPUs
cudaGetDeviceCount(&GPU_N);
...
//declare CUDA streams for each GPU
cudaStream_t stream_halo[GPU_N];
cudaStream_t stream_inner[GPU_N];
for(int n = 0; n<GPU_N; n++)
{
    cudaSetDevice(n);
    //create CUDA streams for each GPU
    cudaStreamCreate(&stream_halo[n]);
    cudaStreamCreate(&stream_inner[n]);
    //enable peer access for each GPU
    for(int m = 0; m<GPU_N; m++)
        if(m != n) cudaDeviceEnablePeerAccess(m,0);
}
...
//time loop
for(int t = 0; t<TMAX; t++)
{
    //exchange halo data
    for(int n = 0; n<GPU_N; n++)
        cudaMemcpyPeerAsync(...,stream_halo[n]);
    for(int n = 0; n<GPU_N; n++)
        cudaMemcpyPeerAsync(...,stream_halo[n]);

    //compute inner nodes
```

```
    for(int n = 0; n<GPU_N; n++){
        cudaSetDevice(n);
        update_inner<<<...,stream_inner[n]>>>(...);
    }

    //synchronize halo stream before computing halo nodes
    for(int n = 0; n<GPU_N; n++)
    cudaStreamSynchronize(stream_halo[n]);

    //compute halo nodes
    for(int n = 0; n<GPU_N; n++)
    {
        cudaSetDevice(n);
        update_halo<<<...,stream_halo [n]>>>(...);
    }
    for(int n = 0; n<GPU_N; n++)
    {
        cudaSetDevice(n);
        cudaDeviceSynchronize();
    }
    swap(A,B);
}
...
return 0;
}
```

4.3 Validation and Performance Evaluation

The GPU code was validated using the benchmark case involving the 3D flow over a square cylinder in channel. This problem was chosen for its simple and well defined geometry, as well as its abundance in simulation results. In this study, the results obtained from the GPU LBM code will be compared against those summarized by Schäfer and Turek [56]. The hardware used for this study was NVIDIA's Tesla M2070, which is a dedicated general purpose GPU. Although this GPU does not have graphics outputs, it has features such as ECC-protected memory and a larger device memory, making them ideal for large scale general purpose calculations. The technical specifications of this GPU are given in Table 4.1. The ECC feature was enabled for all test cases ran in this study. The GPU was connected by PCIe x16 Gen2 to the Intel E5607 CPU at 2.26GHz, running Linux kernel 2.6.32, with CUDA version 5.0 and nvcc release 5.0 version 0.2.1221.

The geometry of the simulation domain consisted of a square cylinder confined in a channel

Table 4.1: Technical Specifications for the NVIDIA Tesla M2070 [47]

Compute Capability	2.0
Peak Single Precision Floating Point Performance	1030 GFLOPS
CUDA Cores	448
Memory Size	6 GB
Memory Bandwidth (ECC off)	150 GB/s

with a square cross section. The height and width of the channel was $4.1d$, where d is the diameter of the cylinder. The cylinder was placed so that its center is $5.0d$ from the inlet, and $2.0d$ from the bottom wall. The total length of the channel is $25.0d$. The four side walls of the channel, as well as the cylinder walls, were set to be no-slip walls. A schematic of the geometry is shown in Figure 4.2. The inflow condition is specified with a prescribed velocity specified as:

$$U_{in} = 16U_m yz(H-y)(H-z)/H^4, V_{in} = 0, W_{in} = 0, \tag{4.1}$$

where H is the height of the channel. The Reynolds number, defined as $Re = \overline{U}d/\nu$, where $\overline{U} = 4/9U_m$, was set to 20, where the flow is expected to be steady. The summary parameters used for validation are $C_D = 2F_x/(\rho\overline{U}^2 dH)$ and $C_L = 2F_y/(\rho\overline{U}^2 dH)$.

Simulations were conducted using a uniform mesh with 10 nodes along the side of the cylinder, with a refinement patch near the cylinder, which increases the grid resolution by a factor of two. Relative to the center of the cylinder, the refinement patch started $2d$ upstream, and ended $4.4d$ downstream, resulting in 128 nodes in the x direction. This value was chosen so that the x dimension of the refinement patch dimension is a multiple of 64. In the y direction, the refinement region spanned $\pm 2d$, relative to the center of the cylinder. In the z direction the entire width of the channel was covered. The block size for computation was set to be $(64 \times 1 \times 1)$.

Similar to the previous chapter where the flow over tandem cylinders were simulated, the moment extrapolation method was used to apply the velocity inflow condition as well as the constant pressure outflow condition. Bounce back conditions were used to model the no-slip walls, and the lift and drag forces were computed based on the momentum exchange method. The characteristic velocity, $\overline{U}$, was set to $0.04c$, and the outlet density was set to 1.0. Simulations were run for $20,000$ time steps, which corresponds to 80 non-dimensional time units ($\tilde{t} = \overline{U}t/d$). The strong scalability of the multi-GPU implementation was also measured by running the simulations on one and two GPUs. Further increasing the

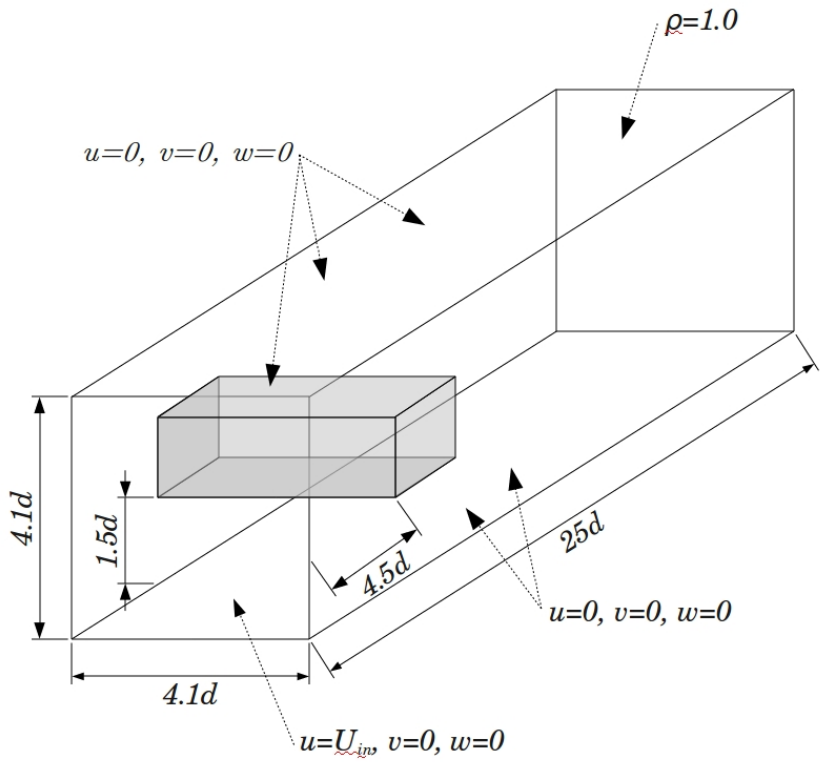

Figure 4.2: Domain geometry and boundary conditions for 3D laminar flow over square cylinder confined in a channel.

Table 4.2: Numerical results for flow over square cylinder at Re=20

	GPUs	C_D	C_L	time [s]	MLUPS
Uniform	1	7.614	0.0658	21	441
	2			12	788
Refinement ×2	1	7.628	0.0690	103	275
	2			56	511
Refinement ×4	1	7.635	0.0693	702	444
	2			358	871
Schäfer and Turek [56]		7.50−7.70	0.06−0.08		

number of GPUs can be achieved by coupling CUDA with OpenMPI, but due to the limited availability of the hardware, this study only concerns the use of up to two GPUs. Table 4.2 summarizes the numerical results obtained, as well as the total run time and execution speed in MLUPS.

Both C_D and C_L were well within the range of values presented in [56]. In terms of performance, the uniform grid implementation on the GPU resulted in impressive execution speeds, with 441 MLUPS and 788 MLUPS for one and two GPUs, respectively. For reference, a serial code on the CPU was implemented, which showed an execution speed of 4.7 MLUPs when run on Intel E5607 CPU at 2.26GHz, compiled with the GNU g++ compiler with O2 optimization. This shows that speedups of two orders of magnitude were obtained from the GPU. The multi-GPU code showed a strong scaling efficiency of over 90% for all three cases, suggesting that the inter-GPU communication is mostly hidden by the computation. The scaling efficiency is expected to improve if a larger domain was used, or if the domain was split in a more sophisticated way that would minimize the amount of communication at the zone interface. In this study however, it was sufficient to prove that the code is capable of managing two GPUs in a reasonably efficient manner. An interesting observation that can be made here is that the execution speed of the case with ×4 refinement is faster than the uniform grid implementation, while the ×2 refinement is considerably slower. It is speculated that compared to the case with the uniform grid, the ×4 refinement case had a considerably larger number of nodes, which lead to a better utilization of the computational resources. The ×2 refinement case on the other hand, suffered in performance, since the additional interpolations and grid interface management routines outweighed the benefits from having more node points to saturate the available compute power.

4.4 Summary

In this chapter, the GPU implementation of the LBM with local grid refinement capabilities was discussed, and tested using a benchmark case. It was found that the current multi-GPU LBM code was capable of accurately reproducing the 3D flow over a square cylinder confined in channel. Speed ups of over 150 times were observed in the GPU code, when compared to a serial CPU implementation. The grid refinement capabilities were also tested in the context of multi-GPU computing, and showed that if the refinement introduces a sufficient number of additional nodes for computation, the performance penalties from grid interface management can be minimized. The strong scalability of the multi-GPU code for up to two GPUs was obtained to be over 90% for both the uniform and non-uniform grid cases.

Chapter 5

Large Eddy Simulations using the Lattice Boltzmann Method

This chapter is concerned about the performance of the LBM for turbulent flow simulations using LES. In the first section, the popular benchmark case of the fully developed turbulent flow in an infinite channel was simulated using a periodic domain. The time averaged velocity and RMS velocities were compared against DNS results, to assess the validity of the LES-LBM code in simulating turbulent flows. The second section is concerned with the application of the LES-LBM code in predicting the flow over a square cylinder confined in a channel. For both cases, the Smagorinsky subgrid scale model with implicit grid filtering was used for turbulence modeling. It should also be noted that in this section, the BGK collision operator was used, since the current MRT implementation introduced some numerical artifacts. The specific cause for these fluctuations are still under investigation. Furthermore, in the context of LES, the grid refinement method that proved to work well under laminar conditions, failed to produce smooth results when turbulence was present. The reason for this is likely due to how the turbulent viscosity in the fine and coarse grids were scaled. A recent study concerning this particular problem was presented by Touil et al. [67].

5.1 Turbulent Channel Flow at $Re_\tau = 180$

5.1.1 Problem Description

The fully developed turbulent flow in a plane channel is one of the most fundamental cases for studying the nature of turbulence, and has been extensively studied both experimentally and numerically. It is an ideal benchmark test case for turbulence models, since DNS data is available for lower Reynolds numbers. For this flow, it is customary to define the flow in terms of the shear Reynolds number, defined as:

$$Re_\tau = \frac{u_\tau \delta}{\nu}, \tag{5.1}$$

where δ is half of the channel height, H, and u_τ is the friction velocity. The friction velocity is related to the wall shear stress, τ_w, by:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}. \tag{5.2}$$

Since the flow is fully developed, the stream-wise forces must balance, as:

$$-\frac{\partial p}{\partial x}\delta = \tau_w. \tag{5.3}$$

Combining Equations 5.2 and 5.3,

$$-\frac{\partial p}{\partial x} = \frac{\rho u_\tau^2}{\delta}. \tag{5.4}$$

The effect of the pressure gradient can be incorporated into the LBM framework by introducing an additional step after the collision step, as [35]:

$$\tilde{f}_i^* = \tilde{f}_i + w_i \delta t \frac{3}{c^2}\frac{\partial p}{\partial x} c_i \cdot \hat{x}. \tag{5.5}$$

For this study, the shear Reynolds number was set to 180. The dimensions of the computational domain were set according to the comprehensive work by Moser et al. [40], with $2\pi H$ in the stream-wise direction, and $\frac{2}{3}\pi H$ in the span-wise direction. In this work, 62 nodes were allocated along H. The upper and lower surfaces of the domain were set to be no-slip by using the bounce back scheme, and the stream-wise and span-wise boundaries were set to be periodic. The standard Smagorinsky model with $C = 0.01$ was used for

the subgrid scale model. The flow was initialized with a uniform stream-wise velocity, and a cubical obstruction was placed to initiate turbulence. The obstruction was removed after several thousand time-steps, and the simulation was run for a total of 3 million time steps. An additional run with 6 million time steps was also run to check for statistical convergence.

5.1.2 Results

The instantaneous flow field is shown in Figure 5.1 in terms of span-wise and stream-wise velocities. Both quantities were normalized based on u_τ. The figure shows that the simulated flow is unsteady with significant 3D flow structures as indicated by the span-wise velocities even after three million time steps, which shows that the turbulence is self-sustaining.

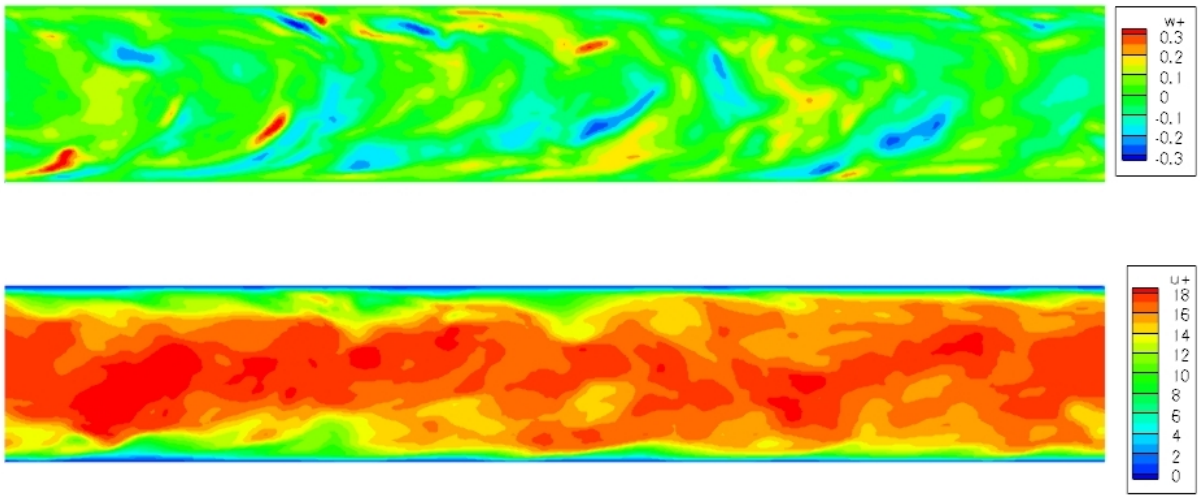

Figure 5.1: Span-wise and stream-wise velocity contours on span-wise cross section. Both velocity components normalized based on u_τ.

The LES-LBM results were quantitatively compared against DNS results presented by Moser et al. [40] in terms of time averaged flow statistics (Figures 5.2-5.3). For the mean stream-wise velocity profile, the LES-LBM simulation predicted a notably lower velocity compared to the DNS results. However, the profile clearly shows the transition from the viscous sublayer to the log-law region, despite the coarse grid (minimum y^+ of 2.2). The

RMS profiles were in reasonable agreement with the DNS results. The peak location of the stream-wise velocity fluctuations showed some discrepancy, but the cross-stream and span-wise velocity components showed very good correspondence. Although it would be interesting to investigate how much the results will improve with a finer grid, the present results were deemed adequate to validate the LES-LBM code in simulating wall-bounded turbulent flow.

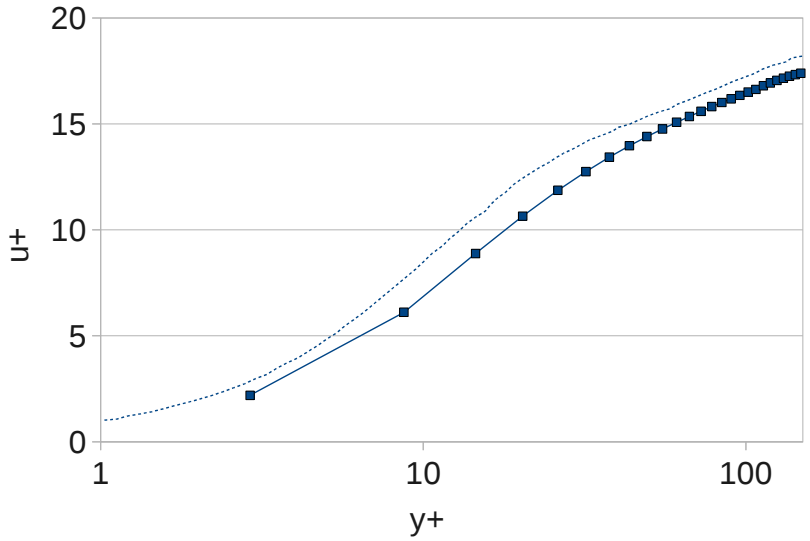

Figure 5.2: Mean stream-wise velocity profile in wall normalized units. Symbols: present work; dotted line: DNS [40].

5.2 Turbulent Flow Over a Square Cylinder in a Channel at $Re_d = 3000$

5.2.1 Problem Description

In the previous section, it was shown that the LES-LBM is capable of simulating the fully developed turbulent channel flow problem. In this section, the LES-LBM was used to predict the flow over a square cylinder confined in a channel. The Reynolds number, defined as $Re = U_{in}d/\nu$, where U_{in} is the inlet velocity, was 3000, and the blockage ratio

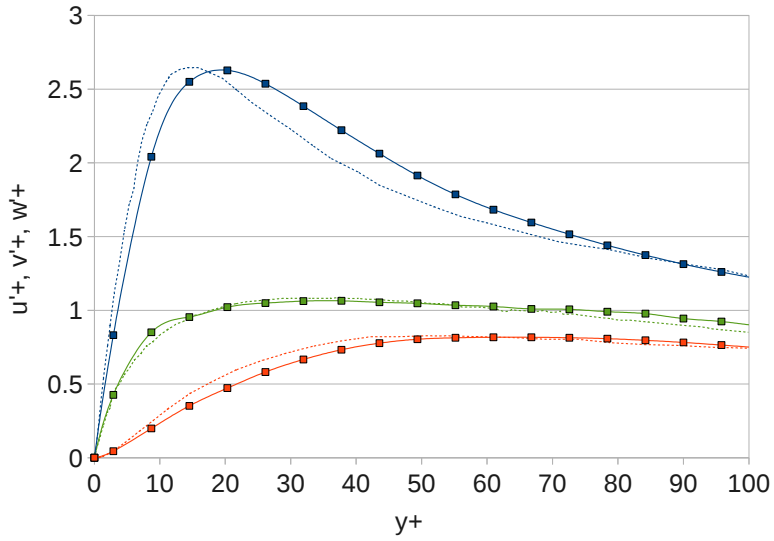

Figure 5.3: RMS velocity profiles in wall normalized units. Blue: u'^{+}; orange: v'^{+}; green: w'^{+}. Symbols: present work; dotted line: DNS [40].

was 20%. This particular flow configuration was studied experimentally by Nakagawa et al. [43], and numerically using a finite volume method based LES by Kim et al. [27].

In the experimental work by Nakagawa et al. [43], the unsteady turbulent velocity field was measured using laser Doppler velocimetry (LDV). The time-averaged and phase-averaged statistics were summarized for various cylinder aspect ratios (width to height). Their experimental setup consisted of a closed water channel with stream-wise, cross-stream, and span-wise dimensions of $30d$, $5d$, and $35d$, respectively. A rectangular cross-sectioned cylinder was installed on the centerline, where the distance between the test section inlet and the front side of the cylinder was $3d$. The flow was generated by a constant head tank, and the water was allowed to pass through a contraction before entering the test section with a mean velocity of U_{in}. The stream-wise and cross-stream velocity components were measured using a two-color four-beamed LDV.

Kim et al. [27] then conducted a numerical study in order to verify the experimental results of Nakagawa et al. [43]. They solved the three-dimensional incompressible Navier-Stokes equations, which were filtered by a box filter, on a non-uniform staggered Cartesian grid, using the finite volume method with the fractional time-stepping method for time integration. Their domain dimensions were made identical to that used in [43], except for the span-wise dimension, which was set to $1d$ with periodic boundary conditions. At the inlet, a uniform mean velocity profile with a thin boundary layer was imposed, and 'jittered' by using random numbers that are 6% of U_{in} in RMS magnitude. A convective condition was employed to model the outlet. For turbulence modeling, a dynamic subgrid scale model, which uses two filters to correlate the subgrid scale stresses with the resolved turbulent stresses [72], was used.

In this study, the LES-LBM was used on a uniform grid, with the standard Smagorinsky model for subgrid closure. The stream-wise and cross-stream dimensions of the domain were set to $20d$ and $5d$, respectively. Two separate cases with span-wise dimensions of $1d$ and $3d$ were conducted, to evaluate its effect on the results. The cylinder location with respect to the inlet location was made identical to [43] and [27] (Figure 5.4). Bounce-back conditions were used to apply no-slip conditions on the channel and cylinder walls, and the outlet was set to have constant pressure. For the inlet condition, instead of prescribing a predetermined velocity profile, a time accurate velocity profile based on a fully developed turbulent channel flow was used. This was done by running an infinite channel flow simulation in parallel with the square cylinder simulation. At each time step, the velocity profile from the channel flow simulation was extracted, and prescribed as an inlet velocity condition for the square cylinder simulation, to allow for an accurate emulation of turbulent inflow in the channel.

To achieve this, the following method was used. For a 3D Cartesian grid, let the nodes be indexed by (i, j, k). Assuming the steam-wise coordinates for the lines B and C in Figure 5.4 are $i = B$ and $i = C$, respectively, the distribution functions at the inlet of the cylinder flow domain can be written as:

$$\boldsymbol{f}(C, j, k) = \boldsymbol{f}^{eq}\left(\rho = \rho(C+1, j, k), \vec{u} = \vec{u}(B, j, k)\right). \tag{5.6}$$

This ensures that the velocity is prescribed based on the channel flow solution, while the density (pressure) is set to be consistent with the cylinder flow domain.

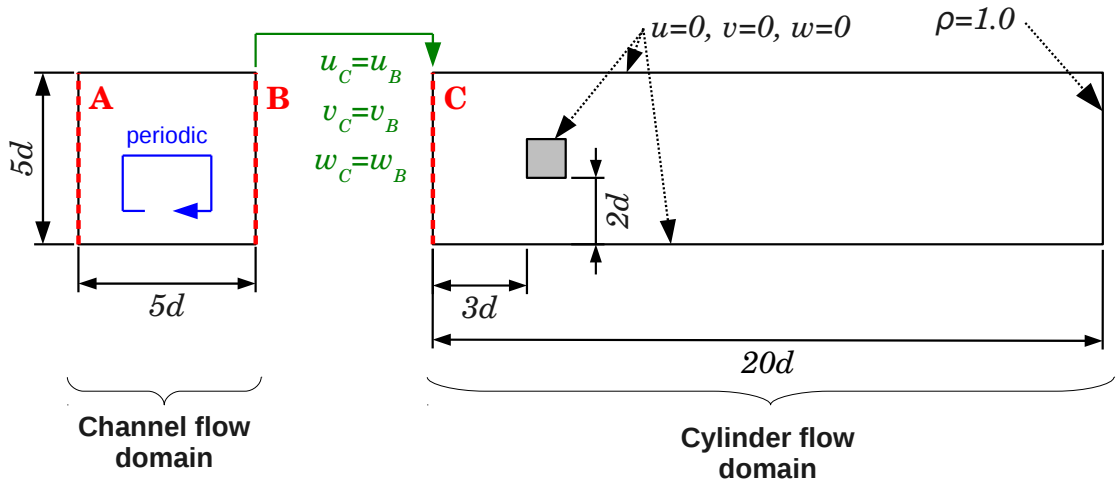

Figure 5.4: Domain geometry and boundary conditions for 3D turbulent flow over square cylinder confined in a channel. Lines A and B indicate the periodic coupling for the infinite channel flow domain, and line C indicates the inlet location for the cylinder flow domain.

5.2.2 Results for Turbulent Channel Flow

Prior to simulating the flow over a cylinder in a channel, the fully developed channel flow at the Reynolds number of interest was studied. Since the Reynolds number based on the diameter of the cylinder is 3000, the equivalent Reynolds number based on the channel height (Re_H) is 15000. The stream-wise dimension was set to H, and the span-wise dimension was set to $0.2H$ and $0.6H$, where $H = 200$ nodes. Turbulence was initiated

by placing a cubical obstacle in the channel, similar to the previous section. The main challenge in this simulation was that since the Reynolds number is specified in terms of the channel height, the shear Reynolds number (Re_τ) is unknown. This means that the pressure gradient required to drive the flow cannot be derived *a priori*. Hence, in this study, a proportional controller, which dynamically adjusts the pressure gradient to achieve the desired flow rate, was used.

The controller was designed to adjust the pressure gradient dynamically, based on the current flow rate in the channel. Letting the current cross-sectionally averaged stream-wise velocity be $u_{current}$, the target stream-wise velocity to be u_{target}, the stream-wise pressure gradient for the next time step to be $\partial p/\partial x|_{next}$, the current stream-wise pressure gradient to be $\partial p/\partial x|_{current}$, and the initial estimate for the stream-wise pressure gradient to be $\partial p/\partial x|_{init}$, $\partial p/\partial x|_{next}$ was adjusted as:

$$\frac{\partial p}{\partial x}|_{next} = \frac{\partial p}{\partial x}|_{current} + \frac{u_{target} - u_{current}}{u_{target}} K_P \frac{\partial p}{\partial x}|_{init} \tag{5.7}$$

where K_P is a constant. In this study, the above correction was applied every 1000 time steps, with K_P set to 0.05.

u_{target} was set to $0.06c$, and the simulation was run for a total of 5 million time steps. The average cross-sectional stream-wise velocity and the applied pressure gradient was plotted for each time step for the $z = 1d$ case (Figure 5.5). It can be seen that the flow rate approaches the target value in the first 2 million time steps, but does not completely converge to a singular value. This behavior was expected, since the turbulent flow is random in nature, and applying a constant pressure gradient does not lead to a constant flow rate. Therefore, to obtain the pressure gradient that achieves the desired average flow rate, the time averaged value of the final 3 million time steps was taken.

In terms of the resulting flow field, it was confirmed that the turbulent flow was self-sustaining, with coherent structures developing in the near-wall regions for both cases (Figure 5.6). Qualitatively, the general flow structures near the wall are similar, but the $z = 3d$ case displayed a higher concentration of the stream-wise vortical structures. This observation is reasonable considering the larger range of wavelengths that are allowed to exist in the span-wise direction for the $z = 3d$ case.

In Figure 5.7, the mean stream-wise velocity profiles for the two cases were plotted, and compared against the log-law profile, defined as:

$$u^+ = \frac{1}{\kappa} ln(y^+) + B, \tag{5.8}$$

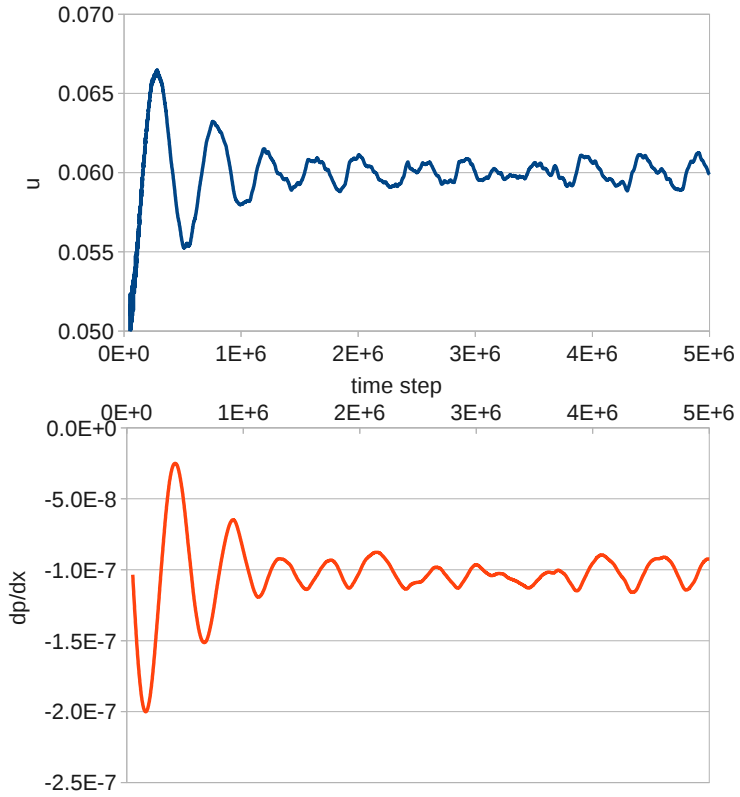

Figure 5.5: Cross-sectionally averaged stream-wise velocity and stream-wise pressure gradient, plotted against time steps.

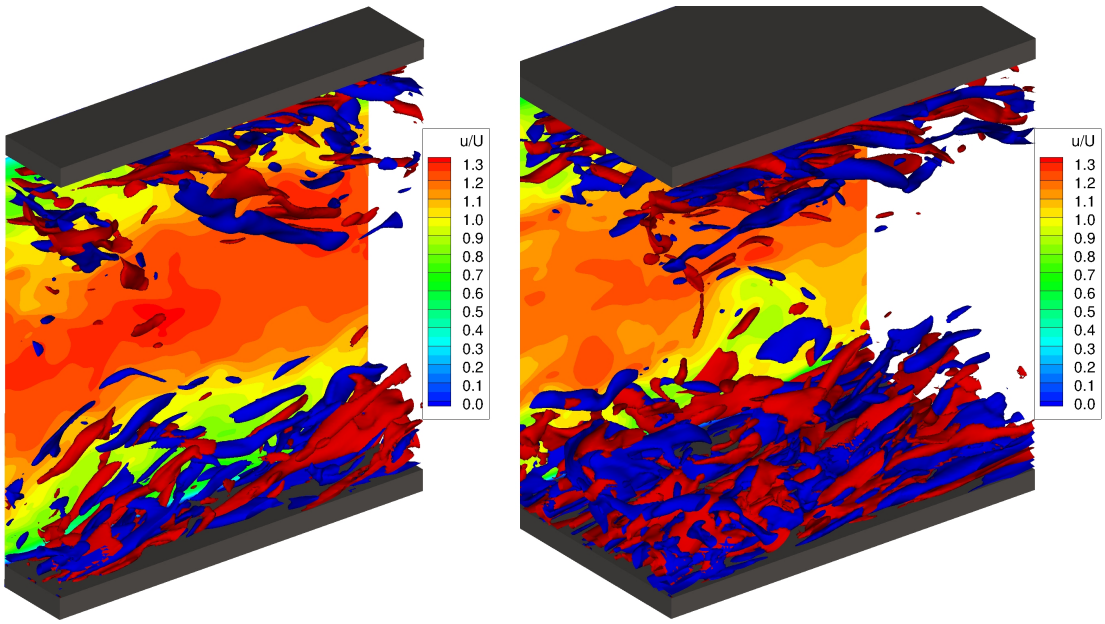

Figure 5.6: Vorticity and velocity plots for fully developed turbulent channel flow at $Re_H = 15000$. Isosurface plots show the stream-wise vorticity (red: $\omega_x = 6.67$; blue: $\omega_x = -6.67$), normalized based on U_{in} and H. Contour plot shows the stream-wise velocity magnitude, normalized based on U_{in}. Left: $z = 1d$; right: $z = 3d$

where $\kappa = 0.41$ and $B = 5.2$. It was found that the $z = 3d$ case showed better agreement to the log-law profile than the $z = 1d$ case. It was interesting to see that although the $z = 1d$ case showed qualitatively acceptable results, the mean velocity profile displayed an obvious discrepancy for $y^+ > 100$, where the velocity gradient rose sharply above the log-law profile.

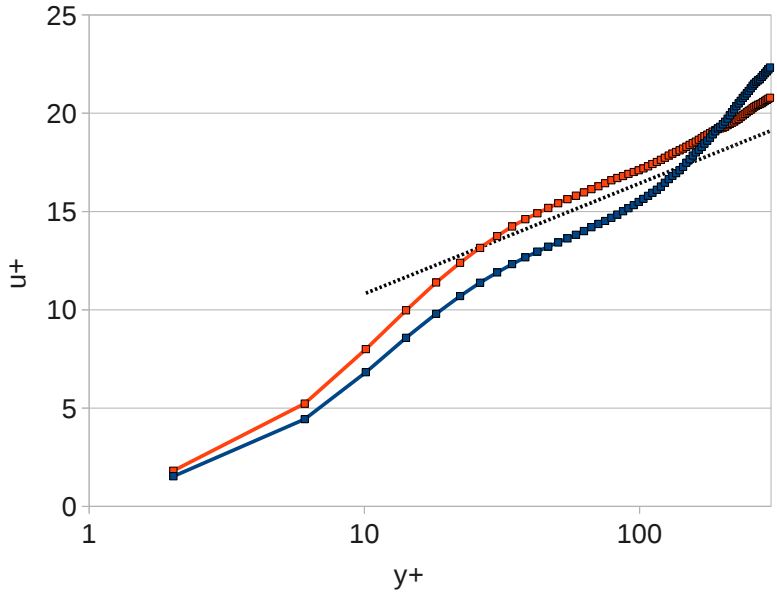

Figure 5.7: Mean stream-wise velocity profiles in wall normalized units. Symbols: present work; dotted line: log-law. Blue: $z = 1d$; orange: $z = 3d$.

5.2.3 Results for Flow Over a Square Cylinder

In the previous section, a dynamic controller was developed to simulate the fully developed turbulent channel flow at $Re_H = 15000$. The resulting flow fields showed reasonable agreement with the log-law, and the pressure gradient required to achieve the desired flow rate was obtained. With this knowledge, it is possible to simulate the flow over a square cylinder confined in a channel, as outlined in Section 5.2.1 (Figure 5.4). Simulations were run for a total of 2 million time steps, where turbulent statistics were collected after 1 million time steps. Similar to the previous section, the effect of the span-wise

dimension was analyzed by setting up two simulations with span-wise dimensions of $1d$ and $3d$. Simulations were also run with a simplified inlet condition, where the inlet velocity is set to U_{in}, to investigate its influence on the results.

The qualitative validity of the present setup was evaluated by observing the instantaneous flow fields. Figure 5.8 shows the instantaneous stream-wise and span-wise velocity components at a span-wise cross section. The channel flow and the cylinder flow domains are connected, to highlight the smooth transfer of the flow conditions at the domain boundary. Careful observation shows that the flow field in the inlet region of the cylinder flow domain is almost identical to that of the inlet region of the channel flow domain, which further validates the consistency of the present inlet boundary condition.

The time averaged flow field was compared against previous studies. Figure 5.9 shows the mean velocity profiles along the wake line of the cylinder. Note that Kim et al. [27] did not present the turbulent statistics along the wake line. From Figure 5.9a, it can be seen that the $z = 1d$ case significantly over-predicted the recirculation length and maximum negative velocity, while the $z = 3d$ case showed good correspondence with both experimental and numerical results. The over-prediction of the recirculation zone in the $z = 1d$ case also affected the stream-wise velocity fluctuations, where it displayed a peak value that is higher by more than 50% of the experimental value (Figure 5.9b). In comparison, the $z = 3d$ case showed only a 10% over-prediction in the peak value. From Figure 5.9c, it was found that both cases showed good agreement with the experimental results. The results obtained from the uniform inlet condition (dotted lines) also showed good correspondence with experimental results. In general, the uniform inlet cases produced better results compared to the $z = 1d$ case with the turbulent inlet, but the $z = 3d$ with the turbulent inlet showed highest correspondence with literature.

The cross-sectional velocity profiles at $1d$, $3.5d$, $6d$, and $8.5d$, were also compared in Figure 5.10. For clarity, only the turbulent inlet condition cases are shown. It was found that the $z = 1d$ case consistently over-predicted the turbulent fluctuations, while the $z = 3d$ case displayed excellent correspondence with experimental results. Comparison with the LES results from Kim et al. [27] indicate that for both the stream-wise and cross-stream fluctuations, the present results show better agreement in the near wake region ($x = 1d$).

The drag and lift force coefficients, as well as the Strouhal numbers, were compared against literature (Table 5.2.3). The table shows that the $z = 3d$ turbulent inlet case was in excellent agreement with experiment [43] in terms of Strouhal number, while the $z = 1d$ uniform inlet case showed the closest correspondence with the LES results by Kim et al. [27]. Considering that Kim et al. [27] enforced the inlet condition as a mean velocity

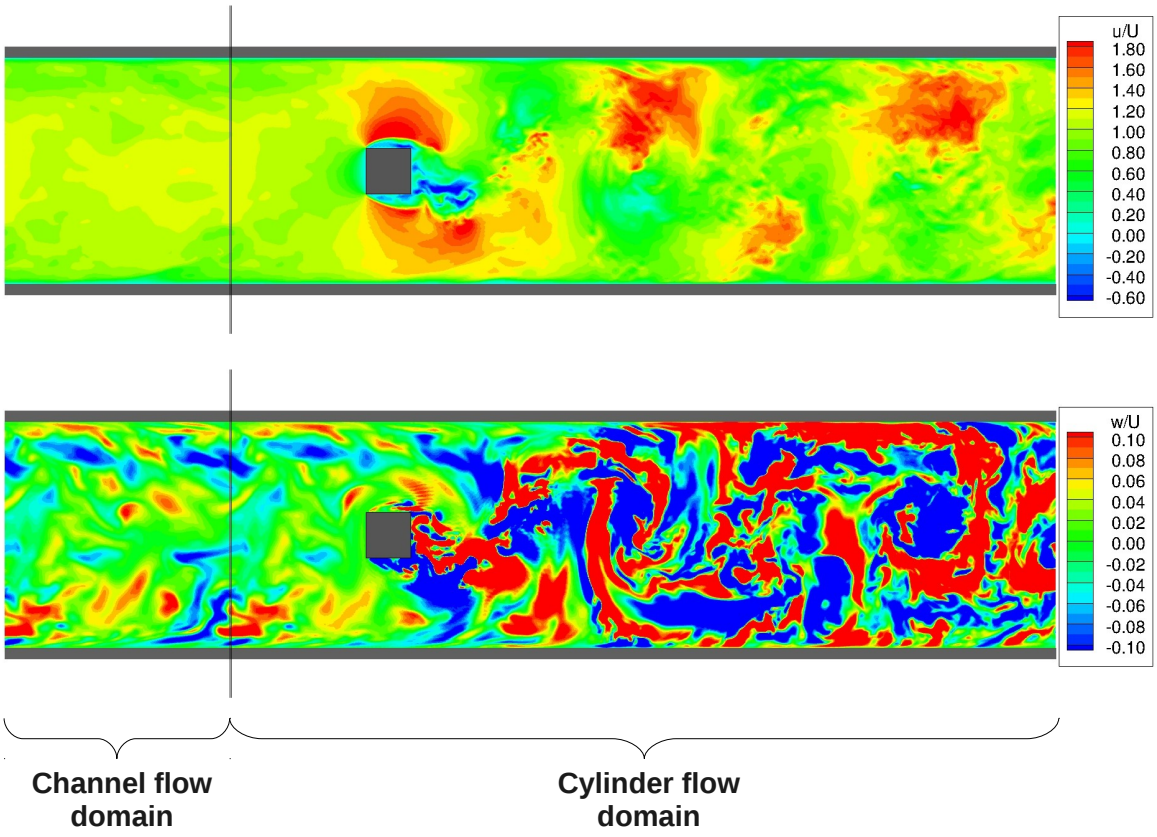

Figure 5.8: Instantaneous stream-wise (top) and span-wise (bottom) velocity components. Black vertical line shows the boundary between the channel flow domain and the cylinder flow domain.

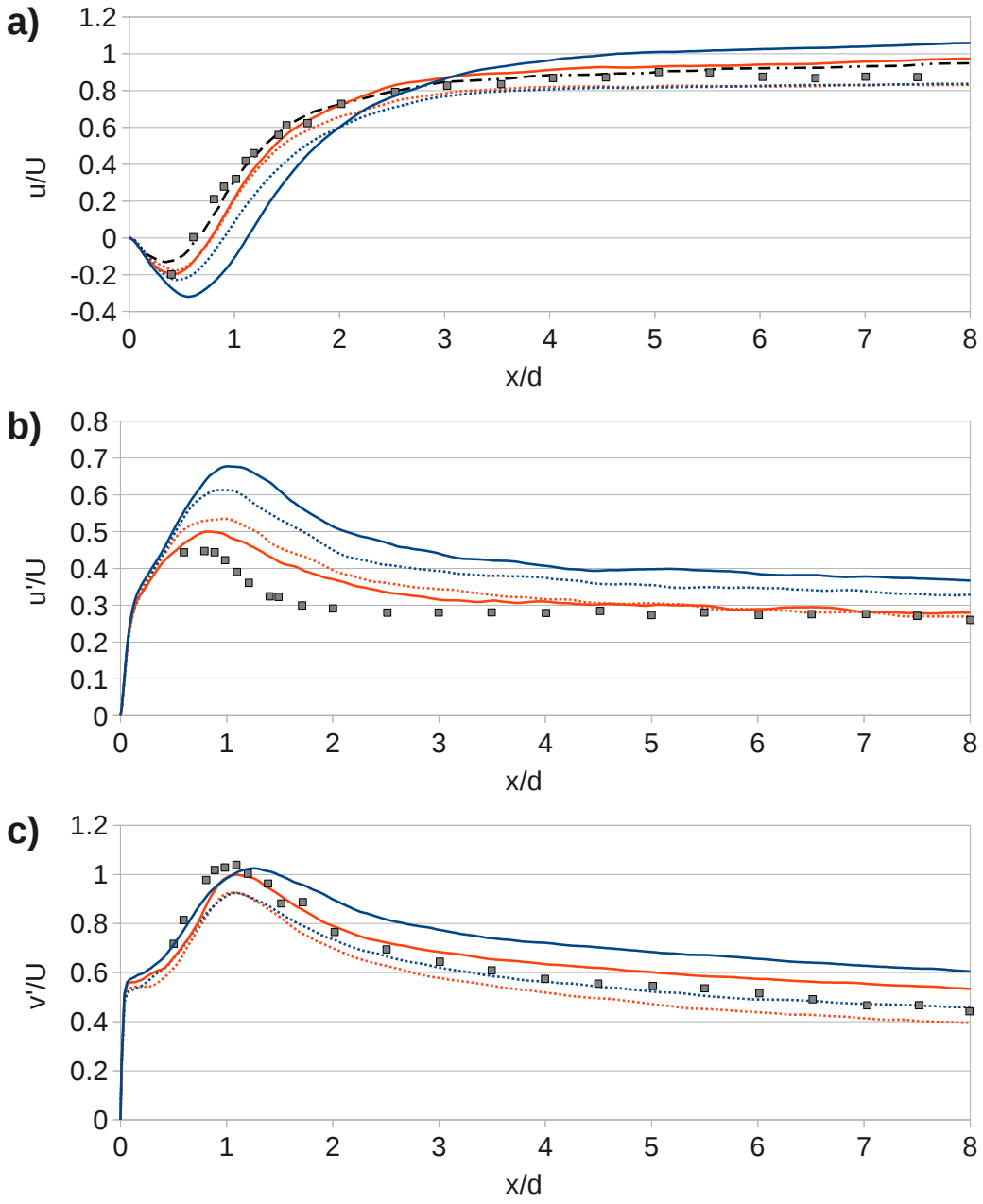

Figure 5.9: Time averaged velocity profiles along the cylinder wake line. Solid lines: present work (turbulent inlet); dotted line: present work (uniform inlet); broken line: LES by Kim et al. [27]; symbols: experimental by Nakagawa et al. [43]. Blue: $z = 1d$; orange: $z = 3d$. a) mean stream-wise velocity; b) RMS stream-wise velocity fluctuations; c) RMS cross-stream velocity fluctuations.

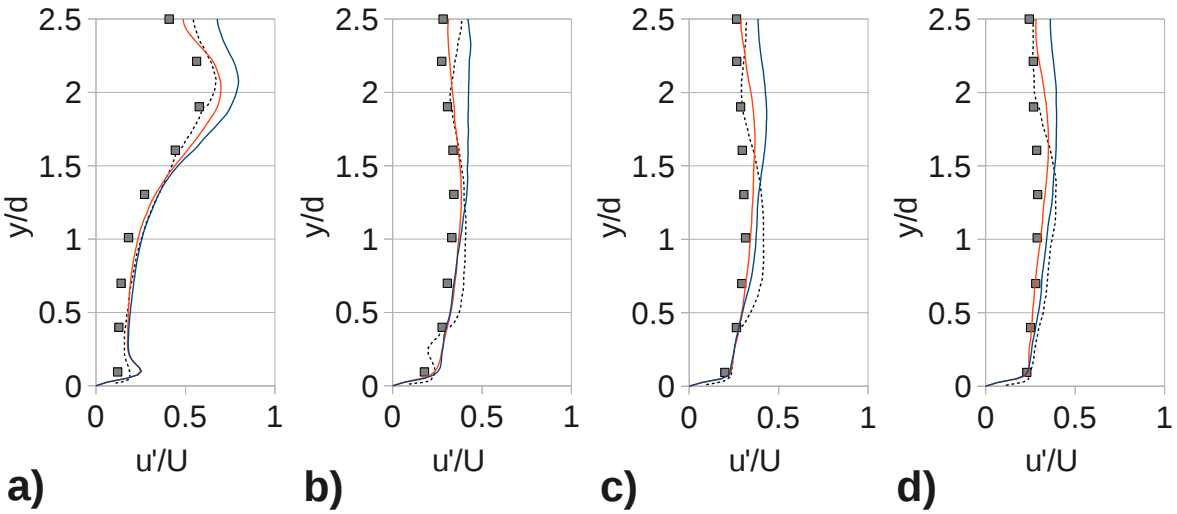

Figure 5.10: RMS stream-wise velocity fluctuation profiles at four cross-stream locations. Solid lines: present work (turbulent inlet); dotted line: LES by Kim et al. [27]; symbols: experimental by Nakagawa et al. [43]. Blue: $z = 1d$; orange: $z = 3d$. a) $x = 1d$; b) $x = 3.5d$; c) $x = 6d$; d) $x = 8.5d$; measured from back face of cylinder.

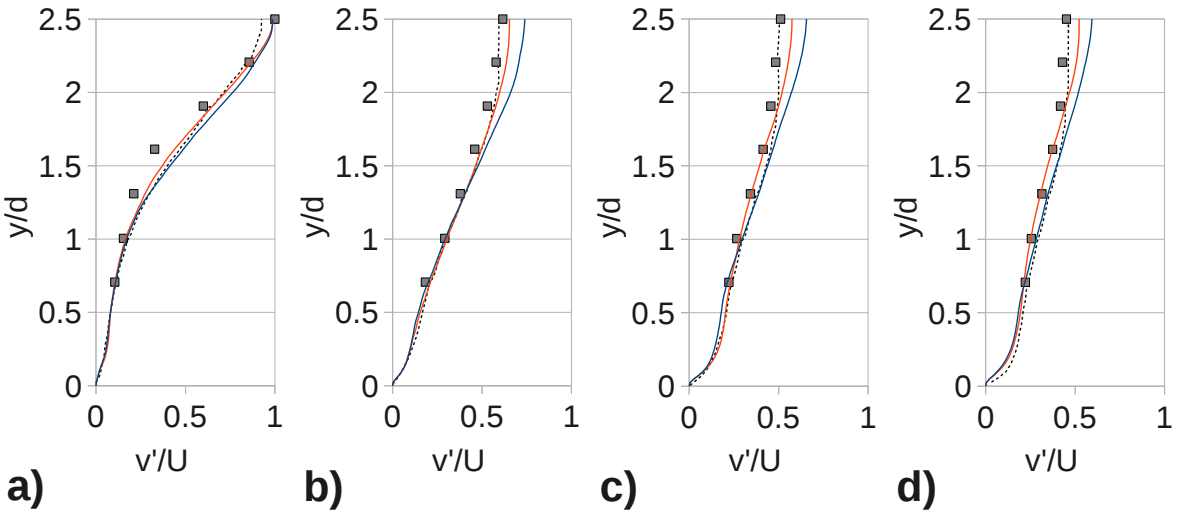

Figure 5.11: RMS cross-stream velocity fluctuation profiles at four cross-stream locations. Solid lines: present work (turbulent inlet); dotted line: LES by Kim et al. [27]; symbols: experimental by Nakagawa et al. [43]. Blue: $z = 1d$; orange: $z = 3d$. a) $x = 1d$; b) $x = 3.5d$; c) $x = 6d$; d) $x = 8.5d$; measured from back face of cylinder.

Table 5.1: Force coefficients and Strouhal numbers for flow over square cylinder in a channel at $Re_d = 3000$

	$\overline{C}_D$	C'_D	C'_L	St
Present				
$z = 1d$, Turbulent inlet	3.40	0.50	1.97	0.136
$z = 3d$, Turbulent inlet	3.10	0.29	1.96	0.133
$z = 1d$, Uniform inlet	2.67	0.48	1.75	0.120
$z = 3d$, Uniform inlet	2.72	0.39	1.87	0.118
Kim et al. [27] (LES)	2.76	0.49	2.06	0.124
Nakagawa et al. [43] (exp.)	-	-	-	0.13

profile with fluctuations created by random numbers, it can be hypothesized based on the present work, that the Strouhal number is significantly affected by the presence of physically coherent turbulent structures at the inlet. It was also observed that the inlet condition and span-wise dimension had profound impact on the drag coefficient, while the lift coefficient was relatively unaffected.

5.3 Summary

In this section, the LBM was used to perform a LES on the GPU. The validity of the LES-LBM was evaluated by first simulating the fully developed turbulent channel flow at $Re_\tau = 180$. The time averaged flow statistics showed good correspondence with DNS results, despite the relatively coarse grid, showing the validity of the LES-LBM for turbulent flow simulations. The code was then used to compute the fully developed turbulent channel flow at $Re_H = 15000$, driven by a dynamically controlled pressure gradient. Good agreement with the log-law profile was observed from the velocity profile, so the channel flow simulation was used to enforce a physically realistic inflow condition for the flow over a square cylinder confined in a channel. The simulation was run with two different span-wise dimensions, as well as with simplified inlet conditions, to quantify its effects on the flow field and force coefficients. The results suggested that the Strouhal number and drag coefficient were significantly affected by the inlet condition and span-wise dimension. It was hypothesized that for simulations involving channel flows, it is important to apply a physically coherent velocity profile that emulates the turbulent boundary layers along the channel walls.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this study, the LBM was applied to both laminar and turbulent flows, and its performance gains from the GPU were analyzed. The primary contributions from this work are:

- A newly developed 3D LBM code was applied to simulate the flow over two tandem cylinders. The effects of the 3D flow structures that develop at low Reynolds numbers ($Re \leq 220$) were quantified by comparing 2D and 3D predictions. It was found that the 3D effects have notable impact on the force coefficients and vortex shedding frequencies. It was also observed that the downstream cylinder promotes the inception of the span-wise flow structures, but the 3D effects experienced by the upstream cylinder are smaller than that of an isolated single cylinder flow.
- The LBM was realized on the GPU using NVIDIA's CUDA. The code was capable of performing LBM simulations with local grid refinements on multiple GPUs. The performance of the GPU code was evaluated using a test case involving the flow over a square cylinder, which showed speedups of over 150 times when two GPUs were used in parallel.
- The GPU code was used to simulate the fully developed turbulent channel flow at $Re_\tau = 180$. The obtained results compared well against DNS results, showing that the LBM is a viable tool for conducting LES.
- The turbulent flow over a square cylinder confined in a channel was also simulated, using a uniform grid. An auxiliary simulation consisting of a periodic channel flow was

used to produce a physically realistic inflow condition for the cylinder flow simulation. Comparisons with numerical and experimental results suggested that accounting for the turbulent structures at the inlet location can significantly improve the results.

6.2 Future Work

Possible future work on this topic include the following:

- Extending the multi-GPU capabilities to allow for GPUs on multiple compute nodes to be used in parallel. The current implementation is restricted to accessing GPUs contained on the same node; combining the current code with MPI can allow access to multiple compute nodes, which can potentially allow for hundreds of GPUs to be used in parallel.
- Turbulence modeling in the current work was restricted to using the Smagorinsky subgrid model, due to its simplicity and suitability for the LBM framework. Future work should investigate the use of wall damping or dynamic subgrid models.
- The LBM is known to be similar to the artificial compressibility method, as it utilizes fictitious compressibility for pressure calculations. A comparison study between the two methods in terms of accuracy and suitability for parallel computing may lead to some interesting findings.

Copyright Permissions

ELSEVIER LICENSE
TERMS AND CONDITIONS

Aug 03, 2013

This is a License Agreement between Yusuke Koda ("You") and Elsevier ("Elsevier") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

Supplier	Elsevier Limited The Boulevard,Langford Lane Kidlington,Oxford,OX5 1GB,UK
Registered Company Number	1982084
Customer name	Yusuke Koda
Customer address	55 Hickory Street East Apt. 404 Waterloo, ON N2J3J5
License number	3198510834459
License date	Jul 29, 2013
Licensed content publisher	Elsevier
Licensed content publication	Computers & Fluids
Licensed content title	Aerodynamic effects of the early three-dimensional instabilities in the flow over one and two circular cylinders in tandem predicted by the lattice Boltzmann method
Licensed content author	Yusuke Koda,Fue-Sang Lien
Licensed content date	30 March 2013
Licensed content volume number	74
Licensed content issue number	None
Number of pages	12
Start Page	32
End Page	43
Type of Use	reuse in a thesis/dissertation
Portion	full article

Format	both print and electronic
Are you the author of this Elsevier article?	Yes
Will you be translating?	No
Order reference number	None
Title of your thesis/dissertation	Lattice Boltzmann Method for Simulating Turbulent Flows
Expected completion date	Aug 2013
Estimated size (number of pages)	
Elsevier VAT number	GB 494 6272 12
Permissions price	0.00 USD
VAT/Local Sales Tax	0.0 USD / 0.0 GBP
Total	0.00 USD

Terms and Conditions

INTRODUCTION

1. The publisher for this copyrighted material is Elsevier. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

GENERAL TERMS

2. Elsevier hereby grants you permission to reproduce the aforementioned material subject to the terms and conditions indicated.

3. Acknowledgement: If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:

Reprinted from Publication title, Vol /edition number, Author(s), Title of article / title of chapter, Pages No., Copyright (Year), with permission from Elsevier [OR APPLICABLE SOCIETY COPYRIGHT OWNER]. Also Lancet special credit - Reprinted from The Lancet, Vol. number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier.

4. Reproduction of this material is confined to the purpose and/or media for which permission is hereby given.

5. Altering/Modifying Material: Not Permitted. However figures and illustrations may be altered/adapted minimally to serve your work. Any other abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of Elsevier Ltd. (Please contact Elsevier at permissions@elsevier.com)

6. If the permission fee for the requested use of our material is waived in this instance, please be advised that your future requests for Elsevier materials may attract a fee.

7. Reservation of Rights: Publisher reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

8. License Contingent Upon Payment: While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by publisher or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received on a timely basis, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

9. Warranties: Publisher makes no representations or warranties with respect to the licensed material.

10. Indemnity: You hereby indemnify and agree to hold harmless publisher and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.

11. No Transfer of License: This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

12. No Amendment Except in Writing: This license may not be amended except in a writing signed by both parties (or, in the case of publisher, by CCC on publisher's behalf).

13. Objection to Contrary Terms: Publisher hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and publisher (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.

14. Revocation: Elsevier or Copyright Clearance Center may deny the permissions described in this License at their sole discretion, for any reason or no reason, with a full refund payable to you. Notice of such denial will be made using the contact information provided by you. Failure to receive such notice will not alter or invalidate the denial. In no event will Elsevier or Copyright Clearance Center be responsible or liable for any costs, expenses or damage incurred by you as a result of a denial of your permission request, other than a refund of the amount(s) paid by you to Elsevier and/or Copyright Clearance Center for denied permissions.

LIMITED LICENSE

The following terms and conditions apply only to specific license types:

15. Translation: This permission is granted for non-exclusive world English rights only unless your license was granted for translation rights. If you licensed translation rights you may only translate this content into the languages you requested. A professional translator must perform all translations and reproduce the content word for word preserving the integrity of the article. If this license is to re-use 1 or 2 figures then permission is granted for non-exclusive world rights in all languages.

16. Website: The following terms and conditions apply to electronic reserve and author websites: Electronic reserve: If licensed material is to be posted to website, the web site is to be password-protected and made available only to bona fide students registered on a relevant course if: This license was made in connection with a course, This permission is granted for 1 year only. You may obtain a license for future website posting, All content posted to the web site must maintain the copyright information line on the bottom of each image, A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or the Elsevier

homepage for books at http://www.elsevier.com , and Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

17. Author website for journals with the following additional clauses:

All content posted to the web site must maintain the copyright information line on the bottom of each image, and the permission granted is limited to the personal version of your paper. You are not allowed to download and post the published electronic version of your article (whether PDF or HTML, proof or final version), nor may you scan the printed edition to create an electronic version. A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx . As part of our normal production process, you will receive an e-mail notice when your article appears on Elseviers online service ScienceDirect (www.sciencedirect.com). That e-mail will include the articles Digital Object Identifier (DOI). This number provides the electronic link to the published article and should be included in the posting of your personal version. We ask that you wait until you receive this e-mail and have the DOI to do any posting.

Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

18. Author website for books with the following additional clauses: Authors are permitted to place a brief summary of their work online only. A hyper-text must be included to the Elsevier homepage at http://www.elsevier.com . All content posted to the web site must maintain the copyright information line on the bottom of each image. You are not allowed to download and post the published electronic version of your chapter, nor may you scan the printed edition to create an electronic version.

Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.

19. Website (regular and for author): A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx. or for books to the Elsevier homepage at http://www.elsevier.com

20. Thesis/Dissertation: If your license is for use in a thesis/dissertation your thesis may be submitted to your institution in either print or electronic form. Should your thesis be published commercially, please reapply for permission. These requirements include permission for the Library and Archives of Canada to supply single copies, on demand, of the complete thesis and include permission for UMI to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission.

21. Other Conditions:

v1.6

If you would like to pay for this license now, please remit this license along with your payment made payable to "COPYRIGHT CLEARANCE CENTER" otherwise you will be invoiced within 48 hours of the license date. Payment should be in the form of a check or money order referencing your account number and this invoice number RLNK501078104. Once you receive your invoice for this order, you may pay your invoice by credit card. Please follow instructions provided at that time.

Make Payment To: Copyright Clearance Center Dept 001 P.O. Box 843006 Boston, MA 02284-3006

For suggestions or comments regarding this order, contact RightsLink Customer Support: customercare@copyright.com or +1-877-622-5543 (toll free in the US) or +1-978-646-2777.

Gratis licenses (referencing $0 in the Total field) are free. Please retain this printable license for your reference. No payment is required.

References

[1] C.K. Aidun and J.R. Clausen. Lattice-Boltzmann method for complex flows. *Annual Review of Fluid Mechanics*, 42:439–472, 2010.

[2] P. Asinari and T. Ohwada. Connection between kinetic methods for fluid-dynamic equations and macroscopic finite-difference schemes. *Computers & Mathematics with Applications*, 58(5):841–861, 2009.

[3] M. Astorino, J.B. Sagredo, and A. Quarteroni. A modular lattice Boltzmann solver for GPU computing processors. Technical report, Mathematics Institute of Computational Science and Engineering, Ecole Polytechnique Fédérale de Lausanne, 2011.

[4] S. Banerjee, K.N. Premnath, and M.J. Pattison. Turbulence simulations using the generalized lattice Boltz-mann equation on massively parallel architectures. In *APCOM07 in conjunction with EPMESC XI, Kyoto, JAPAN*. Citeseer, December 2007.

[5] D. Barkley and R.D. Henderson. Three-dimensional Floquet stability analysis of the wake of a circular cylinder. *Journal of Fluid Mechanics*, 322:215–242, 1996.

[6] M. Bernaschi, M. Fatica, S. Melchionna, S. Succi, and E. Kaxiras. A flexible high-performance lattice Boltzmann GPU code for the simulations of fluid flows in complex geometries. *Concurrency and Computation: Practice and Experience*, 22(1):1–14, 2009.

[7] M. Bouzidi, M. Firdaouss, and P. Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids*, 13:3452–3459, 2001.

[8] B.S. Carmo and J.R. Meneghini. Numerical investigation of the flow around two circular cylinders in tandem. *Journal of Fluids and Structures*, 22(6):979–988, 2006.

[9] B.S. Carmo, J.R. Meneghini, and S.J. Sherwin. Possible states in the flow around two circular cylinders in tandem with separations in the vicinity of the drag inversion spacing. *Physics of Fluids*, 22(5):54101, 2010.

[10] B.S. Carmo, J.R. Meneghini, and S.J. Sherwin. Secondary instabilities in the flow around two circular cylinders in tandem. *Journal of Fluid Mechanics*, 644(1):395–431, 2010.

[11] S. Chen and G.D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30(1):329–364, 1998.

[12] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[13] J. Deng, A.L. Ren, J.F. Zou, and X.M. Shao. Three-dimensional flow around two circular cylinders in tandem arrangement. *Fluid Dynamics Research*, 38(6):386–404, 2006.

[14] D. d'Humières. Multiple–relaxation–time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1792):437–451, 2002.

[15] D. d'Humières and P. Lallemand. Lattice gas automata for fluid mechanics. *Physica A: Statistical Mechanics and its Applications*, 140(1-2):326–335, 1986.

[16] C. Feichtinger, J. Habich, H. Koestler, G. Hager, U. Ruede, and G. Wellein. A flexible patch-based lattice Boltzmann parallelization approach for heterogeneous GPU–CPU clusters. *Parallel Computing*, 37(9):536–549, 2011.

[17] O. Filippova and D. Hanel. Grid refinement for lattice-BGK models. *Journal of Computational Physics*, 147(1):219–228, 1998.

[18] R.K. Freitas, M. Meinke, and W. Schroder. Turbulence simulation via the lattice-Boltzmann method on hierarchically refined meshes. In *European Conference on Computational Fluid Dynamics ECCOMAS CFD 2006*, pages 1–12, 2006.

[19] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the navier-stokes equation. *Physical Review Letters*, 56(14):1505–1508, 1986.

[20] M. Geier, A. Greiner, and J.G. Korvink. Bubble functions for the lattice Boltzmann method and their application to grid refinement. *The European Physical Journal-Special Topics*, 171(1):173–179, 2009.

[21] S. Geller, M. Krafczyk, J. Tölke, S. Turek, and J. Hron. Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows. *Computers & Fluids*, 35(8):888–897, 2006.

[22] I. Ginzbourg and D. d'Humières. Local second-order boundary methods for lattice Boltzmann models. *Journal of Statistical Physics*, 84(5):927–971, 1996.

[23] J. Habich, C. Feichtinger, H. Köstler, G. Hager, and G. Wellein. Performance engineering for the lattice Boltzmann method on GPGPUs: Architectural requirements and performance results. *Computers & Fluids*, pages 266–272, 2012.

[24] J. Habich, T. Zeiser, G. Hager, and G. Wellein. Performance analysis and optimization strategies for a D3Q19 lattice Boltzmann kernel on NVIDIA GPUs using CUDA. *Advances in Engineering Software*, 42(5):266–272, 2011.

[25] S. Hou, J. Sterling, S. Chen, and G.D. Doolen. A lattice Boltzmann subgrid model for high Reynolds number flows. *Pattern formation and lattice gas automata*, 6:151–166, 1994.

[26] W. Jester and Y. Kallinderis. Numerical study of incompressible flow about fixed cylinder pairs. *Journal of Fluids and Structures*, 17(4):561–577, 2003.

[27] D.H. Kim, K.S. Yang, and M. Senda. Large eddy simulation of turbulent flow past a square cylinder confined in a channel. *Computers & Fluids*, 33(1):81–96, 2004.

[28] Y. Koda and F.S. Lien. Aerodynamic effects of the early three-dimensional instabilities in the flow over one and two circular cylinders in tandem predicted by the lattice Boltzmann method. *Computers & Fluids*, 74:32–43, 2013.

[29] M. Krafczyk, J. Tolke, and L.S. Luo. Large-eddy simulations with a multiple-relaxation-time lbe model. *International Journal of Modern Physics B*, 17(1):33–40, 2003.

[30] P. Lallemand and L.S. Luo. Lattice Boltzmann method for moving boundaries. *Journal of Computational Physics*, 184(2):406–421, 2003.

[31] J. Lätt et al. *Hydrodynamic limit of lattice Boltzmann equations*. PhD thesis, Université de Genève, 2007.

[32] J. Li, A. Chambarel, M. Donneaud, and R. Martin. Numerical study of laminar flow past one and two circular cylinders. *Computers & Fluids*, 19(2):155–170, 1991.

[33] K. Mattila, J. Hyväluoma, T. Rossi, M. Aspnäs, and J. Westerholm. An efficient swap algorithm for the lattice Boltzmann method. *Computer Physics Communications*, 176(3):200–210, 2007.

[34] G.R. McNamara and G. Zanetti. Use of the Boltzmann equation to simulate lattice-gas automata. *Physical Review Letters*, 61(20):2332–2335, 1988.

[35] R. Mei, L.S. Luo, and W. Shyy. An accurate curved boundary treatment in the lattice Boltzmann method. *Journal of Computational Physics*, 155(2):307–330, 1999.

[36] J.R. Meneghini, F. Saltara, C.L.R. Siqueira, and J.A. Ferrari. Numerical simulation of flow interference between two circular cylinders in tandem and side-by-side arrangements. *Journal of Fluids and Structures*, 15(2):327–350, 2001.

[37] R. Mittal and S. Balachandar. Effect of three-dimensionality on the lift and drag of nominally two-dimensional cylinders. *Physics of Fluids*, 7:1841–1865, 1995.

[38] S. Mittal, V. Kumar, and A. Raghuvanshi. Unsteady incompressible flows past two cylinders in tandem and staggered arrangements. *International Journal for Numerical Methods in Fluids*, 25(11):1315–1344, 1997.

[39] J. Mizushima and N. Suehiro. Instability and transition of flow past two tandem circular cylinders. *Physics of Fluids*, 17:104107, 2005.

[40] R.D. Moser, J. Kim, and N.N. Mansour. Direct numerical simulation of turbulent channel flow up to Re= 590. *Physics of Fluids*, 11:943, 1999.

[41] A. Mussa, P. Asinari, and L.S. Luo. Lattice Boltzmann simulations of 2D laminar flows past two tandem cylinders. *Journal of Computational Physics*, 228(4):983–999, 2009.

[42] J. Myre, S.D.C. Walsh, D. Lilja, and M.O. Saar. Performance analysis of single-phase, multiphase, and multicomponent lattice-Boltzmann fluid flow simulations on GPU clusters. *Concurrency and Computation: Practice and Experience*, 23(4):332–350, 2011.

[43] S. Nakagawa, K. Nitta, and M. Senda. An experimental study on unsteady turbulent near wake of a rectangular cylinder in channel flow. *Experiments in Fluids*, 27(3):284–294, 1999.

[44] R.R. Nourgaliev, T.N. Dinh, T.G. Theofanous, and D. Joseph. The lattice Boltzmann equation method: theoretical interpretation, numerics and implications. *International Journal of Multiphase Flow*, 29(1):117–169, 2003.

[45] NVIDIA. *NVIDIA CUDA C Best Practices Guide, Version 4.0*, 2011.

[46] NVIDIA. *NVIDIA CUDA C Programming Guide, Version 4.0*, 2011.

[47] NVIDIA. Tesla M-Class GPU Computing Modules, Aug. 2011.

[48] C. Obrecht, F. Kuznik, B. Tourancheau, and J.J. Roux. Multi-GPU implementation of the lattice Boltzmann method. *Computers & Mathematics with Applications*, 2011.

[49] C. Obrecht, F. Kuznik, B. Tourancheau, and J.J. Roux. A new approach to the lattice Boltzmann method for graphics processing units. *Computers & Mathematics with Applications*, 61(12):3628–3638, 2011.

[50] G.V. Papaioannou, D.K.P. Yue, M.S. Triantafyllou, and G.E. Karniadakis. Three-dimensionality effects in flow around two tandem cylinders. *Journal of Fluid Mechanics*, 558(7):387–413, 2006.

[51] S.B. Pope. *Turbulent flows*. Cambridge Univ Pr, 2000.

[52] Kannan N. Premnath, Martin J. Pattison, and Sanjoy Banerjee. Generalized lattice Boltzmann equation with forcing term for computation of wall-bounded turbulent flows. *Physical Review E*, 79(2):026703, 2009.

[53] M.B. Reider and J.D. Sterling. Accuracy of discrete-velocity BGK models for the simulation of the incompressible Navier-Stokes equations. *Computers & fluids*, 24(4):459–467, 1995.

[54] P.R. Rinaldi, EA Dari, MJ Vénere, and A. Clausse. A lattice-Boltzmann solver for 3D fluid simulation on GPU. *Simulation Modelling Practice and Theory*, 25:163–171, 2012.

[55] P. Sagaut. Toward advanced subgrid models for Lattice-Boltzmann-based Large-eddy simulation: theoretical formulations. *Computers & Mathematics with Applications*, 59(7):2194–2199, 2010.

[56] M. Schäfer, S. Turek, F. Durst, E. Krause, and R. Rannacher. Benchmark computations of laminar flow around a cylinder. *Notes on Numerical Fluid Mechanics*, 52:547–566, 1996.

[57] M. Schönherr, K. Kucher, M. Geier, M. Stiebler, S. Freudiger, and M. Krafczyk. Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs. *Computers & Mathematics with Applications*, 61(12):3730–3743, 2011.

[58] B. Sharman, F.S. Lien, L. Davidson, and C. Norberg. Numerical predictions of low Reynolds number flows over two tandem circular cylinders. *International Journal for Numerical Methods in Fluids*, 47(5):423–447, 2005.

[59] C. Shu, N. Liu, and Y.T. Chew. A novel immersed boundary velocity correction–lattice Boltzmann method and its application to simulate flow past a circular cylinder. *Journal of Computational Physics*, 226(2):1607–1622, 2007.

[60] Joseph Smagorinsky. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164, 1963.

[61] M. Stiebler, M. Krafczyk, S. Freudiger, and M. Geier. Lattice Boltzmann large eddy simulation of subcritical flows around a sphere on non-uniform grids. *Computers & Mathematics with Applications*, 61(12):3475–3484, 2011.

[62] D. Sumner. Two circular cylinders in cross-flow: A review. *Journal of Fluids and Structures*, 26(6):849–899, 2010.

[63] M. Thompson, K. Hourigan, and J. Sheridan. Three-dimensional instabilities in the wake of a circular cylinder. *Experimental Thermal and Fluid Science*, 12(2):190–196, 1996.

[64] J. Tölke. Implementation of a lattice Boltzmann kernel using the Compute Unified Device Architecture developed by NVIDIA. *Computing and Visualization in Science*, 13(1):29–39, 2010.

[65] J. Tölke and M. Krafczyk. TeraFLOP computing on a desktop PC with GPUs for 3D CFD. *International Journal of Computational Fluid Dynamics*, 22(7):443–456, 2008.

[66] J. Tölke and M. Krafczyk. Second order interpolation of the flow field in the lattice Boltzmann method. *Computers & Mathematics with Applications*, 58(5):898–902, 2009.

[67] H. Touil, D. Ricot, and E. Lévêque. Direct and large-eddy simulation of turbulent flows on composite multi-resolution grids by the lattice Boltzmann method. *Hyper Articles en Ligne*, 2013.

[68] Xian Wang and Takayuki Aoki. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Computing*, 37(9):521–535, 2011.

[69] C.H.K. Williamson. The existence of two stages in the transition to three-dimensionality of a cylinder wake. *Physics of Fluids*, 31:3165–3168, 1988.

[70] C.H.K. Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28(1):477–539, 1996.

[71] D.A. Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models*, volume 1725. Springer, 2000.

[72] K.S. Yang and J.H. Ferziger. Large-eddy simulation of turbulent obstacle flow using a dynamic subgrid-scale model. *AIAA Journal*, 31(8):1406–1413, 1993.

[73] H. Yu, L.S. Luo, and S.S. Girimaji. LES of turbulent square jet flow using an MRT lattice Boltzmann model. *Computers & Fluids*, 35(8):957–965, 2006.

[74] Y. Zhao. *Modeling natural phenomena with lattice Boltzmann method*. PhD thesis, State University of New York at Stony Brook, 2006.