2010

Lattice Boltzmann modeling for shallow water equations using high performance computing

Kevin Tubbs
Louisiana State University and Agricultural and Mechanical College

LATTICE BOLTZMANN MODELING FOR SHALLOW WATER EQUATIONS USING HIGH PERFORMANCE COMPUTING

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Interdepartmental Program in Engineering Science

by
Kevin Tubbs
B.S. Physics , Southern University, 2001
M.S. Physics, Louisiana State University, 2004
May, 2010

To my family

ACKNOWLEDGMENTS

I want to acknowledge the love and support of my family and friends which was instrumental in completing my degree. I would like to especially thank my parents, John and Veronica Tubbs and my siblings Kanika Tubbs and Keosha Tubbs. I dedicate this dissertation in loving memory of my brother Kendrick Tubbs and my grandmother Gertrude Nicholas.

I would also like to thank Dr. David Constant and Mrs. Claudia Hawkins for the support through my journey here at LSU, my friend Fatima LaJuan Muse for her constant sacrifice and support through difficult times, and my friend Borja Servan Camas for so many valuable discussions and the help he has provided me at many stages.

I am very thankful to my PhD committee members. It has been an honor to me to be able to count with such a committee in my graduate studies.

I want to acknowledge *National Science Foundation GK-12* program and *National Science Foundation IGERT on Multi-scale Computational Fluid Dynamics* for the financial support provided to pursue my doctorate studies at Louisiana State University.

Finally, I want to acknowledge my advisor Dr. Frank T.-C. Tsai for his support, guidance, patience, and confidence in my work.

TABLE OF CONTENTS

DEDICATION.....ii

ACKNOWLEDGEMENTS.....iii

ABSTRACT vi

1 INTRODUCTION 1
1.1 Background 1
1.2 Literature Review 5
1.2.1 Traditional Numerical Methods 5
1.2.2 Lattice Boltzmann Method 8
1.2.3 LBM for Solving Shallow Water Equations 10
1.2.4 LBM on HPC Environments 14
1.3 Objectives of the Study 15
1.4 Goal of the Dissertation 17

2 GOVERNING EQUATIONS 19
2.1 Shallow Water Equations 19
2.2 Multi-layer Shallow Water Equations 21
2.3 Depth-averaged Transport Equation 23

3 LBM FOR SHALLOW WATER EQUATIONS 25
3.1 LBM with BGK Collision Operator 25
3.2 Recovery of Shallow Water Equations in D2Q9 28
3.3 LBM with MRT Collision Operator 30
3.4 LBM for Anisotropic Advection Dispersion 34
3.5 Boundary and Initial Conditions 37
3.5.1 Introduction 37
3.5.2 Periodic Boundary Conditions 37
3.5.3 Solid Boundary Conditions 37
3.5.4 Open Boundary Conditions 39
3.5.5 Initial Conditions 40

4 LBM FOR MULTILAYER SHALLOW WATER EQUATIONS 41
4.1 MRT Collision Operator 41
4.2 Recovery of Multi-layer Shallow Water Equations 43
4.3 Boundary and Initial Conditions 44
4.3.1 Introduction 44
4.3.2 Multilayer Open Boundary Conditions 44
4.4 Multi-Layer LB Algorithm 46

5 HPC LBM FOR SHALLOW WATER EQUATIONS VIA OPENMP 49
5.1 Introduction 49
5.2 Basic Code and Basic Parallelization 49

5.3 Basic Parallelization 50
5.4 Code Optimization and Parallelization 51
5.5 Cache Optimization 53
5.6 Cache Optimization for LBM Using OpenMP 53
5.7 Parallel Speedup and Efficiency 55

6 HPC LBM FOR SHALLOW WATER EQUATIONS VIA GPU ACCELERATION 58
6.1 Introduction 58
6.2 NVIDIA GPU Platform 60
6.3 NVIDIA CUDA 62
6.4 AccelerEye's Jacket 62
6.5 Optimizing MATLAB GPU Performance 63
6.6 Computational Aspects 64
6.7 Parallel Performance 64

7 NUMERICAL EXAMPLES 67
7.1 Dam Break Flow Over A Forward Facing Step 67
7.2 Flow of Partial Dam Break 69
7.3 Mass Transport of Point Continuous Injection 74
7.4 Mass Transport in Partial Dam Break 79
7.5 Circulation in Rectangular Lake 81
7.6 Wind-driven Circulation in Rotating and Non-rotating Basins 85
7.7 Wind- and Density-driven Circulation in Rotating Basins 92

8 CONCLUSIONS 111

REFERENCES 116

VITA 128

ABSTRACT

The aim of this dissertation project is to extend the standard Lattice Boltzmann method (LBM) for shallow water flows in order to deal with three dimensional flow fields.

The shallow water and mass transport equations have wide applications in ocean, coastal, and hydraulic engineering, which can benefit from the advantages of the LBM. The LBM has recently become an attractive numerical method to solve various fluid dynamics phenomena; however, it has not been extensively applied to modeling shallow water flow and mass transport. Only a few works can be found on improving the LBM for mass transport in shallow water flows and even fewer on extending it to model three dimensional shallow water flow fields. The application of the LBM to modeling the shallow water and mass transport equations has been limited because it is not clearly understood how the LBM solves the shallow water and mass transport equations.

The project first focuses on studying the importance of choosing enhanced collision operators such as the multiple-relaxation-time (MRT) and two-relaxation-time (TRT) over the standard single-relaxation-time (SRT) in LBM. A (MRT) collision operator is chosen for the shallow water equations, while a (TRT) method is used for the advection-dispersion equation. Furthermore, two speed-of-sound techniques are introduced to account for heterogeneous and anisotropic dispersion coefficients.

By selecting appropriate equilibrium distribution functions, the standard LBM is extended to solve three-dimensional wind-driven and density-driven circulation by introducing a multi-layer LB model. A MRT-LBM model is used to solve for each layer coupled by the vertical viscosity forcing term. To increase solution stability, an implicit step is suggested to obtain stratified flow velocities. Numerical examples are presented to verify the multi-layer LB

model against analytical solutions. The model's capability of calculating lateral and vertical distributions of the horizontal velocities is demonstrated for wind- and density- driven circulation over non-uniform bathymetry.

The parallel performance of the LBM on central processing unit (CPU) based and graphics processing unit (GPU) based high performance computing (HPC) architectures is investigated showing attractive performance in relation to speedup and scalability.

1 INTRODUCTION

Coastal wetlands make up only a small portion of the United States' land area; however, they are very influential to the economic, social and ecological health of the nation. The loss of coastal wetlands is area of importance to nation and the state of Louisiana. Annual land loss rates in coastal Louisiana have varied over the last 50 years, declining from a maximum of 100 square kilometers (km^2) per yr (39 square miles [mi^2] per yr) for the period 1956–1978. Cumulative loss during this 50-year period in Louisiana represents 80 percent of the coastal land loss in the entire United States (Board 2006). Louisiana accounts for 25 percent of the United States' coastal wetlands and 40 percent of its salts mashes making this issue one of great importance for the state. During colonial times, the contiguous 48 states contained an estimated 221 million acres of wetlands while today about 100 million remain. This loss continues at a rate of 25 miles per year since 1930 (Corel 2004).

Louisiana wetlands are unique and vital ecological assets. Made more evident by the tremendous humanitarian and economic impact of hurricanes Katrina and Rita in 2005, wetlands play in important role in the natural protection of the region from such storms. Wetlands act as both storm buffers and flood control devices during hurricanes and coastal storms. The wetlands also replenish aquifers and purify water by filtering out pollutants and absorbing nutrients as well as provide habitat for a variety of wildlife. Coastal areas in Louisiana also play in important role in shipping for the state and entire nation. Louisiana wetlands and coastal areas are an influential part of the security of the state and the nation.

Coastal wetlands develop due to a balance of natural geomorphologic and coastal ocean processes. These natural processes such as relative sea level rise, wave action, tidal exchange, river discharges, sediment deposition, accumulation of organic material, seawater intrusion and

hurricanes and coastal storms play important roles in the development and sustainability of the wetland. Understanding how these processes interact over time is important to scientist, engineers and policy makers and used to make decisions to ensure the sustainability of the wetland. The same natural processes that develop the wetland also cause the loss of the wetland over centuries. This wetland loss is further complicated human activities. Human activities causing wetland loss include the construction of river levees, large water control structures, and ship and access canals to name a few. Numerical modeling and simulation serves as a valuable tool validate and understand natural processes that affect wetland loss as well as predict and study the effects of human activities on the restoration and management of wetlands in the future. The current trend of increasing computational capabilities allow for more accurate models and more sophisticated management and decision making tools. A major challenge for environmental science is to develop dynamic models that can simulate future environmental responses to the combined effect of human activities and environmental change (J.A Dearing 2006).

This dissertation, though an interdisciplinary and interdepartmental approach, seeks to study and develop new numerical, high-performance-computing modeling tools that would improve management and decision making on Louisiana wetland and coast protection.

1.1 Background

The shallow water equations are used to describe flow in bodies of water where the horizontal length scales are much greater than the fluid depth (i.e., long wavelength phenomena). The shallow water equations have wide applications in ocean engineering, hydraulic engineering (Meselhe et al. 1997; Cao et al. 2004; Zhou et al. 2004; Klar et al. 2008) and coastal engineering (Teeter et al. 2001; Al-Barwani and Purnama 2008; Klar et al. 2008). The shallow water

equations can be used to study main physical phenomena of interest to scientists and engineers such as storm surges (Garcia-Navarro et al. 1992), tidal flows (Banda and Thömmes 2009) and fluctuations in estuary and coastal water regions (Huang and Spaulding 1995; García et al. 2002), tsunami and bore wave propagation (Keming Hu 2006; Simpson and Castelltort 2006), the stationary hydraulic jump, forces acting on off-shore structures, and river, reservoir and open channel flows (Meselhe et al. 1997; Ghidaoui et al. 2001). The shallow water equations can also be coupled to transport equations to model the transport of various physical quantities such as the prediction of pollutant transport in flows (Chertock et al. 2006; Tao and JianHua 2006; Benkhaldoun et al. 2007; Cai et al. 2007), salinity and temperature transport (Loose et al. 2005; P. Ortiz 2006; Navarrina et al. 2008), and sediment transport (Teeter et al. 2001; Wu 2004; Simpson and Castelltort 2006), which are important subjects in many industrial and environmental projects.

The shallow water equations are obtained by assuming a hydrostatic pressure distribution and a uniform velocity profile in the vertical direction. In many cases of practical interests, vertical accelerations of flow are small relative to the horizontal. One can integrate the Navier–Stokes equations along the depth of the fluid body. Then the three-dimensional free boundary problem reduces to a two-dimensional fixed boundary problem with the primary variables being the vertical averages of the horizontal fluid velocities and the fluid depth (Shinbrot 1970; Cobble 1973). When vertical effects are important, for example in baroclinic regimes where density varies with salinity and temperature, the three-dimensional equations should be used. Tan (1992) and Vreugdenhil (1994) provide discussions of shallow water models in both two and three dimensions.

The numerical solution of the shallow water equations is made challenging by a number of factors. The shallow water equations are a system of coupled non-linear partial differential equations defined on complex physical domains arising, for example, from irregular land boundaries. Furthermore, the bottom sea bed (bathymetry) is also often very irregular. Shallow water systems are subjected to a wide variety of external forces, such as the surface wind stress, atmospheric pressure gradient, and tidal potential forces. The Coriolis effect accounts for effect of the Earth's rotation on the shallow water system resulting in an apparent deflection of moving objects when viewed from a rotating reference frame. The Coriolis effect is not a force, however the terms mathematical expression used in the numerical solutions are grouped together with the external forces of the shallow water system. In addition to these physical factors, there are additional difficulties arising from the mathematical nature of the shallow water equations. A major difficulty is the coupling between the fluid depth and the horizontal velocity field which could lead to spurious oscillations or errors if the numerical algorithms are not chosen with care. Due to the fact that viscosity effects, especially horizontal viscosity, are usually relatively small, algorithms that are stable and accurate for smooth to highly advective flows on general geometries are of interest for the numerical solution of these problems.

A substantial literature exists on the application of various finite difference methods, finite volume methods, and finite element methods to the three dimensional shallow water equations (Johnson et al. 1991; Luettich et al. 1991; Lynch and Werner 1991; Casulli and Walters 2000). Each numerical method for shallow water equations has particular advantages and disadvantages. The development and improvement of numerical methods is a current area of research.

1.2 Literature Review

1.2.1 Traditional Numerical Methods

Most numerical algorithms which have been developed for the shallow water equations over the years can be classified into two broad categories. In the first category, the primitive form of the shallow water equations that are obtained from the direct vertical integration of the three-dimensional incompressible Navier–Stokes equations, are numerically solved. In the second category, the primitive shallow water equations are reformulated and the first-order hyperbolic form of the primitive continuity equation is replaced with a second-order wave equation, see. (Lynch and Gray 1979; Luettich et al. 1991). Within those two broad categories, the only difference is the final form of the governing equation. A number of methods have been developed to solve both categories of governing equations. Traditional methods such as the finite difference method (FDM), the finite volume method (FVM) and the finite element method (FEM) are the most used by scientists and engineers.

Finite difference methods (FDM) are commonly being used, such as the Princeton ocean model (POM), the Nearshore Community Model (NearCoM), etc. When using the primitive equation approach to solve the shallow water equations, the use of non-staggered grids in the finite difference context can lead to spurious spatial oscillations (Lynch and Gray 1979). This is also a problem when a straightforward use of equal-order interpolation spaces in the finite element context. Over the years, various researchers have attempted to control these oscillations through the use of staggered grids or mixed interpolation spaces, with limited success. For example, King and Norton (1978) approximated velocities through piecewise quadratic functions and elevations using piecewise linear functions. Johnson et al. (1991) and Blumberg and Mellor (1987) utilized logically rectangular grids with velocities defined on the edges of the elements

and elevation defined at the element centers. Several numerical methods based on the primitive shallow water equations and equal order approximations have also been developed, e.g., Kawahara et al.(1982) , Szymkiewicz (1993), and Zienkiewicz and Codina (1995) who utilized various techniques for controlling oscillations. Zienkiewicz and Ortiz (1995) used a special operator splitting combined with a method of characteristics (MOC). Using these various techniques, these models have been shown to be accurate where the solution is smooth, but not suitable for hyperbolic equations near discontinuities, e.g., shock waves. The advantage of the LBM, shown later, is that it has the ability to be suitable for both.

The reformulation of the primitive equations into a set of hyperbolic equations is necessary to mathematically model discontinuities in the water depth; however, the numerical solution of these equations still remains a challenge. The capability to handle discontinuities was a great challenge and led to the development of shock-capturing high-resolution schemes, in particular, based on the Riemann problem springing between two nodes or elements with a jump in the values of the variable. The development provided the FDM the ability to predict the discontinuous solutions on Cartesian grids (Toro 1992). Another approach in this category is to express the shallow water equations as a system of conservation laws or as advection–diffusion equations if diffusive effects such as eddy viscosity are incorporated. This approach is favored for the FVM and FEM. Aizinger and Dawson (2002) approximated a non-viscid system using a Godunov-type method defined on triangular elements. This approach used discontinuous, piecewise constant approximations of elevation and velocity. A similar method defined on rectangular elements was described by Alcrudo and Garcia-Navarro (1993) for the shallow water equations. In this type of approach, one can make the method ‘‘higher-order’’ through a post-processing step whereby linear terms are added to the solution on each element. Chippada (1998)

tested several different post-processing algorithms and devised a method based on linearizing the system on each element and decoupling the resulting equations. Linear terms (x and y ‘‘slopes’’) were constructed for the elevation variable. Then these slopes were used to enhance the velocity approximation derived from the linearized equations. While this approach gave better accuracy and sharper resolution of the solution for some test problems, it was very ad hoc in nature, and does not always work well in practice. The LBM is capable of handling shocks without the need to solve the Riemann problem of characteristic equations.

Another challenge in solving the shallow water equations is the ability to handle complex geometries. Unstructured meshes are a powerful tool to handle this challenge because they can conform to these various boundaries very easily. This leads to the development of many methods that extended the shock-capturing scheme to the FVM and the FEM. The FVM is very popular due to its simplicity of zero order presentation of elemental unknowns. In order to get a second order scheme, reconstruction such as a MUSCL-like interpolation must be applied (Yoon and Kang 2004). To seek high order accuracy, the spectral volume method promotes the solution order by sub-dividing the spectral cells while keeping the advantages of the normal finite volume method (Wang 2002). This leads to so called well balanced schemes. Later, this study will prove that the LBM with correct forcing terms is well balanced and capable complex geometry.

One extension of the shock-capturing method to the FEM could be the discontinuous Galerkin (DG) finite element method, which is getting popular recently. Cockburn (2003) made a series studies on the discontinuous Galerkin method on general differential equations. Unlike the usual continuous FEM, which usually assembles a global system and solves a huge linear set, the DG FEM lies between the FVM and the FEM. It has the advantages of locally enforced mass conservation (element by element) and of the ability to capture steep gradients and fronts. The

DG FEM does not limit itself on the selection of element basis pairs indicating compatibility of velocity and pressure, which is important and troublesome in the continuous FEM. The flux continuity through element interfaces can be weakly enforced by the commonly used approximate Riemann solvers, which particularly are the Harten Lax and Van leer approach (HLL) (Harten et al. 1983), the Roe approximate Riemann solver (Roe and Balsara 1996), etc. It has been found that the limiter plays an important role in suppressing the unphysical oscillations in high order methods. Most limiters come from the idea in one-dimensional case that no local extremer is created during the interpolation (Cockburn 2003). The LBM has been compared to DG FEM solutions with favorable results .

1.2.2 Lattice Boltzmann Method

The lattice Boltzmann method (LBM) is an alternative numerical scheme for simulating a wide range of fluid dynamics and transport phenomena. The LBM was originally created to model flows governed by the Navier-Stokes equations (Gunstensen et al. 1991; Alexander et al. 1992; Chen et al. 1992; Chen and Doolen 1998; He et al. 1998; Inamuro et al. 1999; Lallemand and Luo 2000; Wolf-Gladrow 2000). More recently, the LBM has developed into an alternative and promising numerical technique for a wide range of computational fluid dynamics (CFD) techniques (Dawson et al. 1993; Martinez et al. 1994; He et al. 1998; Kang et al. 2002; Kang et al. 2002). This method can be either regarded as an extension of the lattice gas automata or as a special discrete form of the Boltzmann equation from the kinetic theory of gases. The method is based on statistical physics and models the fluid flow by tracking the evolution of the distribution functions of the fluid particles in phase space. The method can be considered in the class of kinetic theory approaches.

The LBM is based on solving the discrete-velocity Boltzmann equation in statistical physics as opposed to conventional numerical schemes based on the discretization of partial differential equations describing macroscopic conservation laws. It describes the microscopic picture of the movement of particles in an extremely simplified way, while at the macroscopic level, it gives a correct average description. The essential idea of the LBM approach lies in the recovery of the macroscopic governing equations, e.g., the Navier-Stokes equation, the shallow water equation, the diffusion equation, the advection-diffusion equation, etc., from the microscopic flow behavior of the particle movement as described by kinetic theory. The approach does not use the actual details of the particles but follow a collection of fictitious particles whose properties recover the macroscopic behavior. The basic idea is to replace the nonlinear differential equations of macroscopic fluid dynamics with a simplified description modeled on the kinetic theory of gases. To obtain the hydrodynamic behavior, the Chapman-Enskog expansion, which is a perturbation expansion in time and space to describe slowly varying solutions of the underlying kinetic equations, is undertaken. The advantages of the method are its ease in parallelization because of the locality of particle interaction and the transport of particle information, and flexibility in geometry because of the easy implementation of complex boundary conditions and complex properties of a fluid system. The method has been proven to be effective in exploiting these advantages in various applications and implementations on different high performance computing architectures. Furthermore, the method has become an alternative to conventional numerical methods like FDMs, FEMs, and FVMs in computational fluid dynamics.

The LBM has found a wide range of applications in a variety of fields including the shallow water equations. The LBM has been successfully adopted to simulate shallow water

equations of wind-driven ocean circulation (Salmon 1999; Zhong et al. 2006), to model three-dimensional planetary geostrophic equations (Salmon 1999b) and to study atmospheric circulation of the northern hemisphere with ideal boundary conditions (Feng et al. 2001). Many free surface flows can be modeled by the shallow water equations with the assumption that the vertical scale is much smaller than the horizontal scale. These equations can be derived from the depth-averaged incompressible Navier-Stokes equations. The application fields of shallow water equations include a wide spectrum of phenomena in environmental and hydraulic engineering, including tidal flows in an estuary or coastal regions, rivers, reservoir and open channel flows.

1.2.3 LBM for Solving Shallow Water Equations

Modeling of problems in hydrodynamics, hydraulics, and environmental fluid mechanics may be undertaken at three different length scales, commonly referred to as the microscopic, mesoscopic, and macroscopic levels (Frisch et al. 1986). Microscopic modeling involves the application of Newton's laws to every molecule in the system. It requires knowledge of the initial state of each molecule and the quantification for the interactions among all the molecules in the system. Because of the level of detail needed, microscopic modeling is computationally infeasible except in some cases where the mean free path between molecules is large. Mesoscopic modeling (i.e., modeling from statistical perspectives) entails the application of Newton's laws to a probability distribution of molecules. Mesoscopic modeling uses the Boltzmann equation as a starting point for system simulation, where the dependent variable is the probability distribution of particles (Reitz 1981). Mass, momentum, energy, and entropy are computed from the moments of this distribution function. Macroscopic modeling entails the application of the basic laws of mechanics and thermodynamics to a continuum. Examples of the macroscopic continuum models in hydraulic engineering, hydrodynamics, and environment fluid

mechanics include the shallow water equations (Tubbs and Tsai 2008; Tubbs and Tsai 2009), Richard's unsaturated flow equation (Zhang et al. 2002; Ginzburg 2006), the Navier-Stokes equations, and the equation of chemical species transport (Chen et al. 1993; Dawson et al. 1993; Deng et al. 2001).

Until recently, the analyses and solutions of problems in hydraulics, hydrodynamics, and environmental fluid mechanics have been based exclusively on macroscopic continuum models, which are solved either analytically or numerically. However, over the last three decades numerical schemes based on mesoscopic models have been developed and applied to a multitude of hydrodynamic problems, including shock waves in compressible flows (Chu 1965; Reitz 1981; Xu et al. 1995; Xu et al. 1996), multicomponent and multiphase flows (Gunstensen et al. 1991; Xu 1997; He et al. 1998), flows in complex geometries (Rothman 1988; Chen and Doolen 1998), turbulent flows (Chen et al. 1992; Martinez et al. 1994), low Mach number flows (Su et al. 1999), and heat transfer and reaction diffusion flows (Qian 1993; Xu 1999). Mesoscopic models based on the Boltzmann equation can be categorized into two sub-classes: continuous Boltzmann models and discrete Boltzmann models such as the LBM. The main difference in the two is that the distribution function in is continuous or discrete in particle velocity, respectively. Excellent reviews of the LBM and continuous Boltzmann models are provided by Chen and Doolen (1998), Xu (1999), and Ghidaoui et al. (2001).

In those reviews, the main advantages of mesoscopic Boltzmann based numerical models over macroscopic based numerical models are summarized below:

1. While the advective operator in the macroscopic approach is non-linear, its counterpart in the mesoscopic approach is linear (Chen and Doolen 1998).

2. A mesoscopic based numerical model can be easily extended to multidimensional flows because the distribution function of particles is a scalar (Xu et al. 1996).
3. In mesoscopic modeling, the implementation of complex boundary conditions is straightforward (Reitz 1981; Frisch et al. 1986; Abbott and Minns 1998; Chen and Doolen 1998).
4. In mesoscopic modeling, the incompressible flow solution is obtained in the limit as the Mach number tends to zero. This means that the solutions of two and three-dimensional non-hydrostatic surface water models do not involve the tedious and difficult solution of the Poisson equation for the pressure field (Su et al. 1999).
5. The scalar nature of the Boltzmann distribution function and the fact that the Boltzmann equation is only the first-order ordinary differential equation (ODE) give mesoscopic modeling the intrinsic features required for parallel computation (Abbott and Minns 1998; Chen and Doolen 1998). This is highly beneficial for direct numerical simulation (DNS) and large eddy simulation (LES) of turbulent open channel flows.
6. The diffusion and viscous terms that appear as second derivative terms in macroscopic modeling are represented by a simple algebraic difference term in mesoscopic modeling. Thus, the need for separate treatment of the advection and diffusion terms is eliminated.
7. The collision function in mesoscopic models eliminates the need for numerical entropy fixes to ensure that the second law of thermodynamics is not violated by the solution (Xu et al. 1995). In contrast, macroscopic numerical models require *ad hoc* entropy fixes in order to satisfy the entropy condition (Reitz 1981; Xu et al. 1995).

The fact that mesoscopic numerical models satisfy the entropy condition was exploited by Prendergast and Xu (1993) and Xu et al. (1995) to model shock waves in compressible flows and

by Gunstensen et al. (1991), Shan and Doolen (1995), Xu (1997), and He et al. (1998) to model interfaces in multiphase and multi-component flows. Ghidaoui et al. (2001) reports that these applications revealed that the mesoscopic approach both accurately resolves shocks and discontinuities, and does not suffer from the failures associated with the Riemann solution of macroscopic hydrodynamics equations. The failures of Riemann solvers are well documented in Roberts (1990), Einfeldt et al. (1991) and Quirk (1998).

The many attributes of mesoscopic modeling, particularly its success in resolving shocks in compressible flows and resolving interfacial discontinuities in multiphase flow, suggest that mesoscopic modeling may be useful in simulating fluid flows in shallow water regimes. In these problems, hydraulic jumps may occur, requiring a method that is flexible to resolve both the flow regimes and the shocks. The LBM approach is applied to one and two-dimensional shallow water flows and extended to three-dimensional shallow water flows through a multi-layer approach. Numerical experiments show that the LBM based shallow water model produces accurate results for rapidly and gradually varied open channel flow problems and does not suffer from non physical oscillations as encountered when applying Riemann solvers to the macroscopic equations. This finding is consistent with conclusions reported by researchers using Boltzmann theory to model shock waves in compressible flows (Pullin 1980; Reitz 1981; Xu et al. 1995; Xu et al. 1996) and shallow water flows (Salmon 1999; Salmon 1999b; Zhou 2002; Zhong et al. 2006; Zhou 2007). Moreover, mesoscopic based numerical models are simpler to formulate, apply, and implement, including high performance computing environments, than Riemann solvers since they do not require characteristic decomposition.

1.2.4 LBM on HPC Environments

The LBM has achieved success in the world of computational physics and has also been used in graphics and visualization for simulating a variety of fluid phenomena with complex boundary conditions (Thurey and Rude 2004; Wei et al. 2004; Chu and Tai 2005; Han et al. 2007; Zhao et al. 2007). These areas have been dominated by the traditional FDM, FVM and FEM, which have their advantages and disadvantages. All are capable of being implemented on high performance computing architectures with different performance benefits and limitations. Comparing the performance of different computational methods is always a difficult task. Since the established FDM, FVM and FEM are the result of an evolution over many decades, one might expect that the simple LBM cannot compete. The accuracy and performance of the lattice-Boltzmann method have been compared to those of FDM (Noble et al. 1996; Sankaranarayanan et al. 2003), FVM (Bernsdorf et al. 1999; Breuer et al. 2000; Geller et al. 2006) and FEM (Chen et al. 1992; Martinez et al. 1994; Kandhai et al. 1999; Geller et al. 2006). These various studies have confirmed that the lattice Boltzmann method is competitive with other approaches. Indeed, it is faster in situations where a specified accuracy is required, in particular in the context of the time-dependent simulation of large, complex systems by means of parallel implementations. However, a comparison between different fluid solvers is prone to ambiguity since their accuracy, intrinsic speed and convergence behavior all depend on the chosen parameters and specific details of the implementation. Implementation of LBM on CPU-based systems is a current source of research and numerous improvements are possible starting from standard LBM implementations (Succi 2001; Wilke et al. 2003; Pohl et al. 2004; Wellein et al. 2006). The implementation of LBM on CPU-based architectures is achieved on both distributed and shared memory systems. LUDWIG (Desplat et al. 2001) is a parallel LBM code for fluids,

implementing message passing interface (MPI) to achieve full portability and good efficiency on both massively parallel processors (MPP) and symmetric multiprocessing (SMP) systems. With OpenMP (Board 2008), the LBM has been optimized to implement on multiple CPUs with shared-memory parallel programming (Bella et al. 2002).

More recently, the LBM has been seen as a good candidate for implementation on hardware accelerated systems using Graphics Processing Units (GPU). It has been accelerated on a single GPU (Li et al. 2005; Zhao 2008; Tubbs and Tsai 2009; Tubbs and Tsai 2010) or a GPU cluster (Fan et al. 2004) with MPI. Moreover, the LBM for the Navier-Stokes equations was implemented in two dimensions using the Compute Unified Device Architecture (CUDA™) interface developed by NVIDIA®. Nevertheless, all these applications use a programming style close to the hardware especially developed for graphics applications.

1.3 Objectives of the Study

The numerical simulating of large systems leads to the necessity of largely increasing the computational capability available. Currently, the main trend to increment this computational capability is based on clustering CPUs to operate in parallel rather than on increasing CPUs processing speeds. Hence the suitability of a numerical scheme to be parallelized is becoming an important feature to be considered. In this framework LBM offers a great capability to be parallelized based on its explicit nature and locality, which results in high scalability performance.

On one hand, LBM is still under development and the reason is because LBM is barely two decades old, which makes it a relatively new numerical method in comparison with traditional methods like FDM, FEM and FVM. First developed to model the Navier-Stokes equation, the majority of literature on LBM has been focused on purpose. Recently, LBM is

gaining attention to model various partial differential equations with applications in a wide range of engineering and science disciplines. Only over the past decade, LBM has become an attractive alternative for modeling hydrodynamic and transport problems in many areas such as ocean, hydraulic and coastal engineering (e.g. shallow water flows). Since LBM is still at a disadvantage with respect to its competitors regarding solving complex problems, more research has to be put on developing the theoretical basis and implementation techniques to make LBM capable of coping with complicated problems and become a practical tool in modeling shallow water equations.

1. This study aims to investigate the use of the LBM for shallow water equations and the anisotropic advection-dispersion equation with velocity-dependent dispersion coefficients on HPC environments. The choice of collision operator in both the shallow water and advection-dispersion equation play an important role in the stability and accuracy of the method.
2. This study aims to compare the MRT collision operator to the SRT (BGK) collision operator for the shallow water equations at the situation that the relaxation time parameter is close to the stability limit of 0.5. The MRT collision operator is selected in order to increase stability and accuracy and eliminate spurious oscillations when the BGK model fails.
3. This study aims to demonstrate that the speed-of-sound techniques are capable of account for the heterogeneity and anisotropy in the dispersion coefficient in mass transport in shallow water. Specifically, the speed-of-sound techniques are capable of coping with the discontinuous free surface water depth in the transport problem.

4. This study aims to extend the standard LB model for the simulation of three-dimensional shallow water flows using a multi-layer LB model. A MRT-LBM is used to solve each layer coupled by the vertical viscosity forcing term. To increase solution stability, an implicit step is suggested to obtain flow velocities.
5. This study aims to demonstrate the proposed multi-layer LB model by testing the influence of wind stress, vertical viscosity forcing, bottom friction and bathymetry. The numerical results of flow velocities for wind-driven and density-driven circulation in a rectangular lake with flat bottom and non-uniform bathymetry are investigated.
6. This study aims to investigate the parallel performance of the multi-layer LB model using OpenMP. The parallel decomposition along only on the horizontal flow directions has two advantages: 1.) It retains the inherent parallelism of the LBM for each layer; and 2.) It retains the locality of the tridiagonal solver over layers with respect to threads. The study has shown that the use of explicit loop control is important in maintaining linear speedup as the number of processors increase.
7. This study aims to investigate the parallel performance of the LBM on GPU-based HPC environments using MATLAB code and the Jacket GPU engine. Moreover, the study aims to investigate how the parallel performance scales with the problem size. Due to the architecture of the GPU, the performance increases with increasing computational intensity and decreasing need for communications between sub-domains.

1.4 Goal of the Dissertation

This dissertation aims to investigate the implementation of the LBM formulation for the shallow water flow and mass transport equations on both CPU based and GPU based architectures. Optimization of the LBM formulation on CPU-based systems exhibits promising

performance by exploiting the inherent parallelism of LBM and selecting a good memory access pattern that uses CPU cache efficiently. A hardware-accelerated LBM shallow water formulation is promising because of the easy coding of LBM and straightforward GPU mapping, allowing it to achieve excellent computation efficiency up for large data sets. The current trend in high performance computing is the use of heterogeneous computer systems which will require methods that perform well on CPU-based systems and accelerators like GPU-based systems. Due to the features of the LBM, it has the potential to increase the performance of various applications in fluid flow and transport in shallow water systems using CPU-based or GPU-based systems.

The main purpose of this dissertation is to extended the standard to lattice Boltzmann method (LBM) for shallow water flows to deal with three dimensional flow fields coupled to mass transport, investigate the stability and accuracy of the method, and investigate its performance on high performance computing (HPC) environments.

2 GOVERNING EQUATIONS

2.1 Shallow Water Equations

Consider a shallow water flow regime shown in Figure 2.1. Due to the fact that the horizontal length scale is much greater than the vertical length scale, the shallow water equations are derived by depth-integrating the continuity equation and the Navier-Stokes equations. The depth integration of the mass transport equation leads to the shallow water transport equation. The shallow water equations with forcing terms of wind, bottom friction, bed slope and the term representing the Coriolis effect are given as (Zhou 2004):

Continuity Equation:

$$\frac{\partial h}{\partial t}+\frac{\partial\left(hu_i\right)}{\partial x_i}=0 \tag{2.1.1}$$

Momentum Equations:

$$\frac{\partial\left(hu_i\right)}{\partial t}+\frac{\partial\left(hu_iu_j\right)}{\partial x_j}+\frac{\partial}{\partial x_i}\left(\frac{gh^2}{2}\right)=\nu\left[\frac{\partial^2\left(hu_i\right)}{\partial x_j\partial x_j}\right]+F_i \tag{2.1.2}$$

where i and j are Cartesian indices and the Einstein summation convention is used, h is the water depth, u_i is the depth-averaged velocity component in the i direction, t is the time. The forcing terms are given as:

$$F_i=F_{Pi}+F_{bi}+F_{Wi}+F_{Ci} \tag{2.1.3}$$

$$F_{Pi}=-gh\frac{\partial z_b}{\partial x_i} \tag{2.1.4}$$

$$F_{bi}=C_bu_i\sqrt{u_iu_i} \tag{2.1.5}$$

$$F_{Wi}=\frac{\rho_a}{\rho_w}C_Wu_{Wi}\sqrt{u_{Wi}\cdot u_{Wi}} \tag{2.1.6}$$

$$F_{Ci} = \begin{cases} f_c h u_y, & i = x \\ -f_c h u_x, & i = y \end{cases} \tag{2.1.7}$$

where z_b is the bed elevation, $g = 9.81\, m/s^2$ is the gravitational acceleration , ν is the kinematic viscosity and F_i, is the external force acting on the shallow water flow consisting of the hydrostatic pressure approximation, F_{Pi}, the bed shear stress, F_{bi}, the wind shear stress, F_{wi}, and the forcing term representing the Coriolis effect, F_{Ci}. $f_c = 2\varpi \sin\varphi$ is the Coriolis parameter and ϖ is rotation rate of the earth and φ is the latitude, z_b is the bed elevation, $C_b = g/C_z^2$ is the bed friction coefficient and $C_z = h^{1/6}/n_b$ is the Chezy coefficient given with the Manning coefficient at the bed, n_b, $[L^{-1/3}T]$. ρ_w is the density of water, ρ_a is the density of air, $C_W = (0.63 + 0.66\sqrt{u_{Wi} \cdot u_{Wi}}) \times 10^{-3}$ is the wind coefficient, and u_{Wi} is the wind velocity in the i direction.

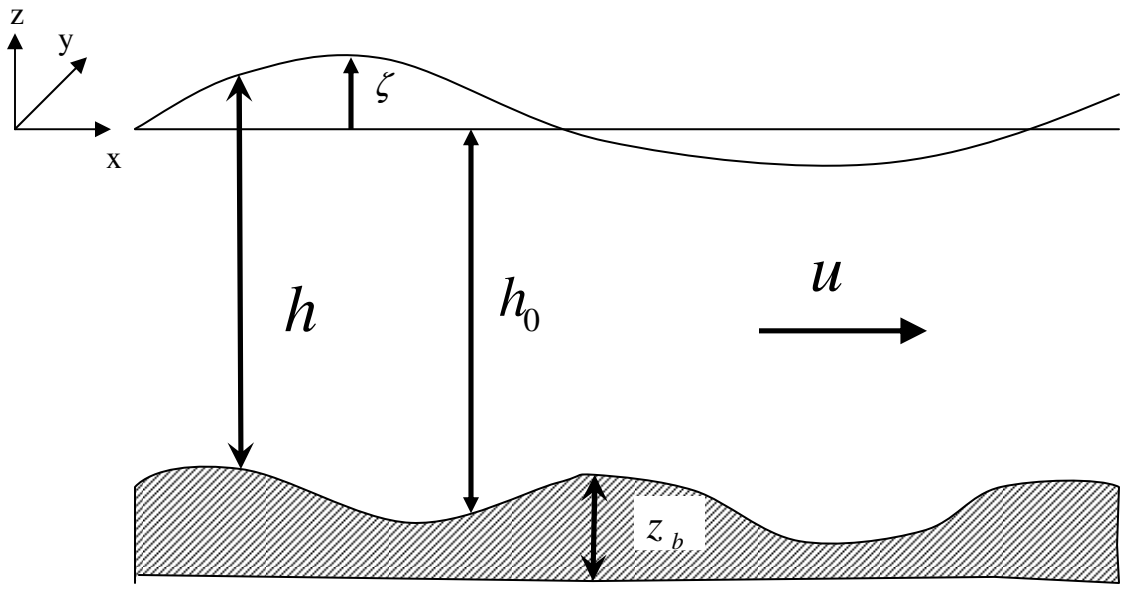


Figure 2.1: Shallow water flow regime

2.2 Multi-layer Shallow Water Equations

Although flows in coastal and estuarine areas are usually classified as shallow water flows, details of the vertical structure of the flow is necessary for better modeling. There is a need to simulate three-dimensional free surface flows. This would require the solution of a system of equations coupling the Navier-Stokes equation to a moving free surface boundary. This approach is computationally expensive and may have difficulties handling the discontinuities in the free surface. Consider the multi-layer discretization of shallow water flow illustrated in Figure 2.2. The multi-layer shallow water equations under the hydrostatic assumption present an alternative solution to the free surface Navier-Stokes system and lead to a precise description of the vertical profile of the horizontal velocity while preserving the robustness and computational efficiency of the shallow water equations. Based on the multi-layer Saint-Venant system (Audusse 2005; Audusse et al. 2006), the governing equations are similar to the traditional shallow water equations with additional terms for transferring momentum between the layers:

$$\frac{\partial h^{(\ell)}}{\partial t}+\frac{\partial\left(h^{(\ell)}u_i^{(\ell)}\right)}{\partial x_i}=0 \tag{2.3.1}$$

$$\frac{\partial\left(h^{(\ell)}u_i^{(\ell)}\right)}{\partial t}+\frac{\partial\left(h^{(\ell)}u_i^{(\ell)}u_j^{(\ell)}\right)}{\partial x_j}+\frac{\partial}{\partial x_i}\left(\frac{1}{2}gh^{(\ell)}\sum_{m=1}^{M}h^{(m)}\right)=\nu\left[\frac{\partial^2\left(h^{(\ell)}u_i^{(\ell)}\right)}{\partial x_j\partial x_j}\right]+F_i^{(\ell)} \quad ,\ell=1,2,\cdots,M \tag{2.3.2}$$

where $h^{(\ell)}$ is the local water height in layer ℓ, $u_i^{(\ell)}$ is the local velocity component in the i direction in layer ℓ, $F_i^{(\ell)}$ is the external force acting on layer ℓ, g is the gravitational acceleration, ν is the kinematic viscosity, x_i is the Cartesian coordinate, and t is time. M is the total number of layers. The external force consists of the wind-driven forcing term ($F_{Wi}^{(\ell)}$) (only

for the top layer), the bed slope forcing term ($F_{Pi}^{(\ell)}$), the vertical kinematic eddy viscosity term ($F_{\mu i}^{(\ell)}$), the non-conservative pressure source term ($F_{NCi}^{(\ell)}$) (Audusse 2005; Audusse et al. 2006; Audusse and Bristeau 2007; Audusse et al. 2008), the density gradient forcing term, $\left(F_{\rho}^{(\ell)}\right)$ (Shankar et al. 1997), and the forcing term representing the Coriolis effect ($F_{Ci}^{(\ell)}$) as follows

$$F_i^{(\ell)} = F_{Wi}^{(\ell)} + F_{Pi}^{(\ell)} + F_{\mu i}^{(\ell)} + F_{NCi}^{(\ell)} + F_{Ci}^{(\ell)} + F_{\rho}^{(\ell)} \tag{2.3.3}$$

$$F_{Wi}^{\ell} = \delta_{M\ell}\frac{\tau_{iz}^{W}}{\rho} = \delta_{M\ell}\frac{\rho_a}{\rho}C_W U_{Wi} W_s \tag{2.3.4}$$

$$F_{Pi}^{(\ell)} = -gh^{(\ell)}\frac{\partial z_b}{\partial x_i} \tag{2.3.5}$$

$$F_{\mu i}^{(\ell)} = -\kappa\delta_{1\ell}u_i^{(\ell)} + 2\mu\left(1-\delta_{M\ell}\right)\frac{u_i^{(\ell+1)}-u_i^{(\ell)}}{h^{(\ell+1)}+h^{(\ell)}} - 2\mu\left(1-\delta_{1\ell}\right)\frac{u_i^{(\ell)}-u_i^{(\ell-1)}}{h^{(\ell)}+h^{(\ell-1)}} \tag{2.3.6}$$

$$F_{NCi}^{(\ell)} = -\frac{gH^2}{2}\frac{\partial}{\partial x_i}\left(\frac{h^{(\ell)}}{H}\right) \tag{2.3.7}$$

$$F_{\rho}^{(\ell)} = \int_{\zeta}^{\overline{z}}\frac{\partial\rho}{\partial x}dz \tag{2.3.8}$$

$$F_{Ci}^{(\ell)} = \begin{cases} f_c h^{(\ell)}u_y, & i = x \\ -f_c h^{(\ell)}u_x, & i = y \end{cases} \tag{2.3.8}$$

where $\delta_{1\ell}$ and $\delta_{M\ell}$ are Kronecker delta functions; $f_c = 2\varpi\sin\varphi$ is the Coriolis parameter and ϖ is the rotation rate of the Earth and φ is the latitude; $\overline{z} = \frac{h^{(\ell)}}{2} + \sum_{m=1}^{\ell-1}h^{(m)}$ is the location of the center of a layer, z_b is the bed elevation, τ_{iz}^{W}, is the wind stress in the i direction, ρ is the fluid density, ρ_a is the air density, C_W is the wind stress coefficient, U_{Wi} is the wind velocity

components in the i direction, $W_s = |U_{Wi}|$ is the wind speed measured $10\,m$ above the water surface, κ is the bottom friction coefficient, and μ is the vertical (kinematic) eddy viscosity. The water depth is the sum of local water heights of all layers, i.e. $H = \sum_{m=1}^{M} h^{(m)}$. The free surface elevation above the datum $z = 0$, is the water depth minus the still water depth, H_0, i.e. $\zeta = H - H_0$.

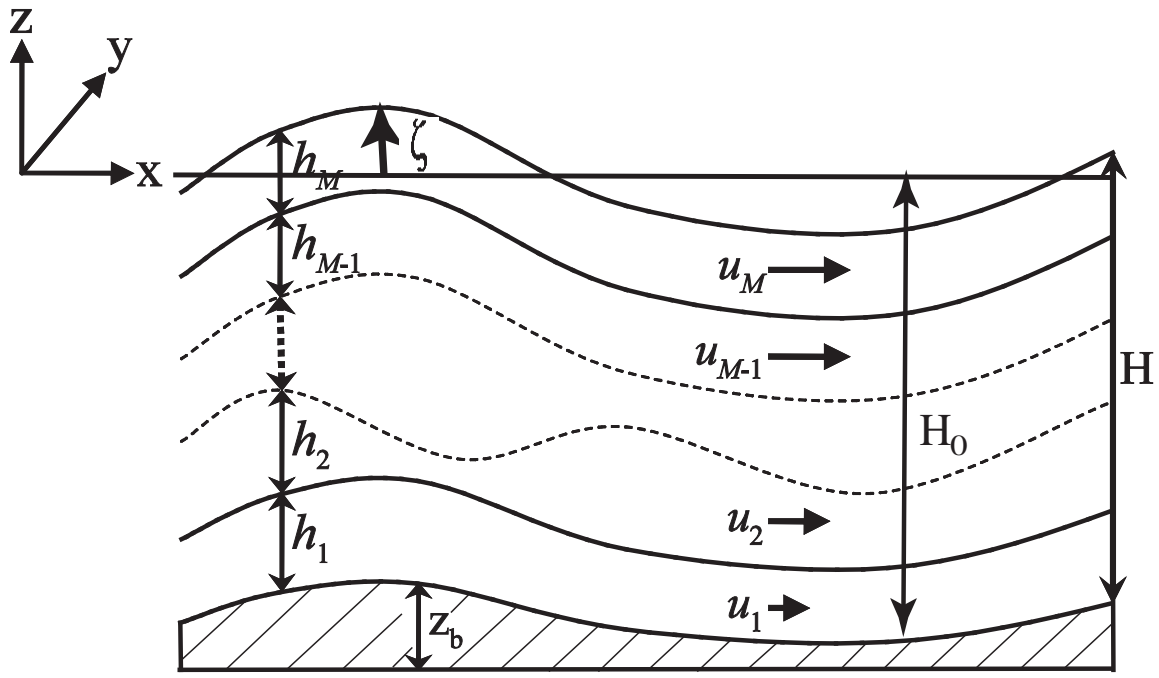


Figure 2.2. Multi-layer shallow water flow.

2.3 Depth-averaged Transport Equation

The two-dimensional depth averaged anisotropic advection dispersion equation (AADE) are (Tao and JianHua 2006).

$$\frac{\partial(hC)}{\partial t} + \frac{\partial(hu_i C)}{\partial x_i} = \frac{\partial}{\partial x_i}\left(D_{ij} h \frac{\partial C}{\partial x_j}\right) + S_C \tag{2.3.9}$$

where

$$S_C = -KhC + S_0 h \tag{2.3.10}$$

$$D_{ij} = \left(k_L |\mathbf{u}| \delta_{ij} + (k_L - k_T) \frac{u_i u_j}{|\mathbf{u}|} \right) \frac{h\sqrt{g}}{C_z} \tag{2.3.11}$$

C is the concentration, K is an attenuation coefficient, S_0 is the concentration source term, δ_{ij} is the Kronecker delta, and D_{ij} is the eddy dispersion tensor. As the principal directions of anisotropic eddy dispersion do not align with the flow directions, a distinct dispersion coefficient is defined for each Cartesian direction. Furthermore, the flow direction may not align with one Cartesian direction and therefore a general eddy dispersion tensor must be used and is defined in equation (2.3.10) (Elder 1959; Tao and JianHua 2006), where k_L and k_T are the longitudinal and transverse coefficients, which are dimensionless.

3 LBM FOR SHALLOW WATER EQUATIONS

3.1 LBM with BGK Collision Operator

The LBM was first developed to solve the equations of hydrodynamics governed by the Navier-Stokes equation based on the kinetic theory of gases described by the Boltzmann equation (McNamara and Zanetti 1988). The discrete Boltzmann equation for describing dynamics of local particle distribution functions in a discrete velocity field is

$$\frac{\partial f_\alpha}{\partial t} + c_{\alpha i}\frac{\partial f_\alpha}{\partial x_i} = \Omega_\alpha \tag{3.1.1}$$

where f_α is the particle distribution function moving along α direction; f_α^{eq} is the equilibrium distribution function (EDF), λ is the relaxation time, $\mathbf{c}_\alpha = \{c_{\alpha i}\}$ is the streaming velocity along α direction, $\Omega_\alpha = -\left(f_\alpha - f_\alpha^{eq}\right)/\lambda$ is the Bhatnagar-Gross-Krook (BGK) collision operator (Bhatnagar et al. 1954) which represents changes in f_α due to particle collisions.

The lattice Boltzmann equation is obtained by integrating Eq. (3.1.1) in time along the α direction. In each time step, the particle distribution functions arrive to their neighboring nodes at the same time through prescribed lattice connections. Therefore, the streaming velocity $\mathbf{c}_\alpha$ along α direction is not arbitrary and is determined by the lattice connection and size. Figure 3.1 gives a diagram of the two dimensional lattice with nine discrete velocities, known as the D2Q9 lattice. The streaming velocity for D2Q9 is

$$\mathbf{c}_\alpha = \begin{cases} (0,0) & \alpha = 0 \\ c\left[\cos\left(\frac{1}{4}(2\alpha-2)\pi\right), \sin\left(\frac{1}{4}(2\alpha-2)\pi\right)\right] & \alpha = 1,2,3,4 \\ \sqrt{2}c\left[\cos\left(\frac{1}{4}(2\alpha-9)\pi\right), \sin\left(\frac{1}{4}(2\alpha-9)\pi\right)\right] & \alpha = 5,6,7,8 \end{cases} \tag{3.1.2}$$

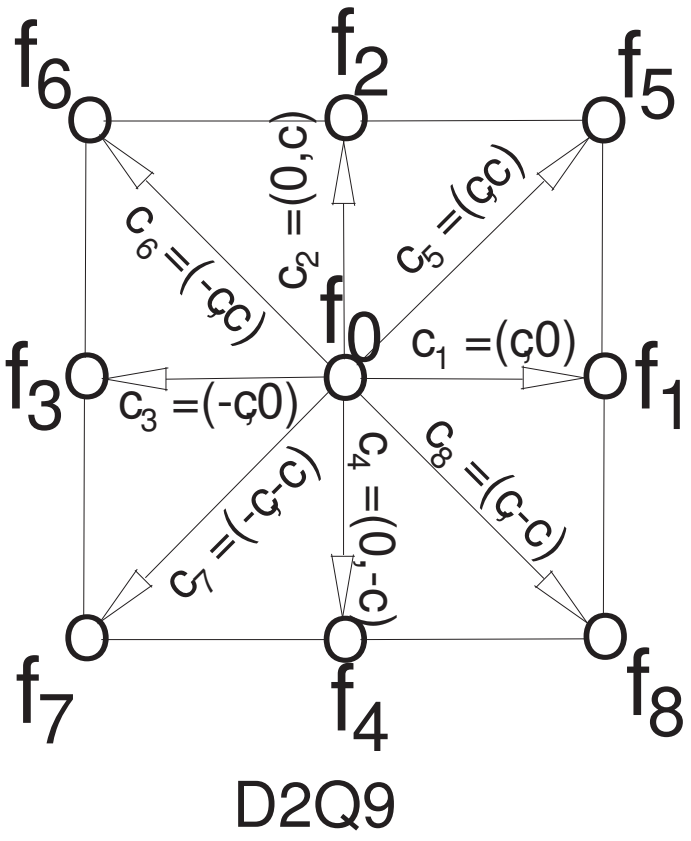


Figure 3.1. D2Q9 lattice. D stands for the number of dimensions and Q stands for the number of lattice velocities.

The lattice Boltzmann equation with BGK collision operator (LBGK) on a D2Q9 lattice for shallow water is given as follows (Salmon 1999; Zhou 2002; Zhou 2007):

$$f_\alpha\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t\right) = f_\alpha\left(\mathbf{x},t\right) - \frac{1}{\tau}\left(f_\alpha\left(\mathbf{x},t\right) - f_\alpha^{\,eq}\left(\mathbf{x},t\right)\right) + \frac{\Delta t c_{\alpha i}}{6c^2} F_i\ \left(\mathbf{x},t\right) \tag{3.1.3}$$

where, $\tau = \lambda/\Delta t$ is the relaxation time parameter, $c = \Delta x/\Delta t$ is the lattice speed, Δx is the lattice size, Δt is the time step and the third term in equation (3.1.3) is the forcing term representing the external forces in equation (2.1.3). The choice of discretization of the forcing term is determined in the recovery of the shallow water equations presented in section 3.2. The macroscopic variables of water depth and flow velocity are calculated as the zeroth and first moments of the distribution functions, respectively:

$$h = \sum_\alpha f_\alpha \tag{3.1.4}$$

$$hu_i = \sum_\alpha c_{\alpha i} f_\alpha \tag{3.1.5}$$

The EDFs are given by

$$\begin{aligned} f_\alpha^{eq} &= h\omega_\alpha \left(\frac{c_s^2}{c^2} + \frac{\mathbf{u}\cdot\mathbf{c}_\alpha}{c^2} + \frac{3}{2}\frac{\left(\mathbf{u}\cdot\mathbf{c}_\alpha\right)^2}{c^4} - \frac{1}{2}\frac{\mathbf{u}\cdot\mathbf{u}}{c^2} \right) \quad ,\alpha > 0 \\ f_\alpha^{eq} &= h - \sum_{\alpha>0} f_\alpha^{eq} \end{aligned} \tag{3.1.6}$$

where $c_s^2 = gh/2$ is the squared speed of sound, and ω_α are the weighting factors that depend on lattice directions and the type of lattice to be used. For D2Q9, $\omega_\alpha = 1/3 \quad ,\alpha = 1,2,3,4$ and $\omega_\alpha = 1/12 \quad ,\alpha = 5,6,7,8$. The EDFs are derived to satisfy the following constraints on the zeroth, first, second and third moments:

$$\sum_\alpha f_\alpha^{eq} = h \tag{3.1.7}$$

$$\sum_\alpha c_{\alpha i} f_\alpha^{eq} = hu_i \tag{3.1.8}$$

$$\sum_\alpha c_{\alpha i} c_{\alpha j} f_\alpha^{eq} = h\left(\delta_{ij} c_s^2 + u_i u_j\right) \tag{3.1.9}$$

$$\sum_\alpha c_{\alpha i} c_{\alpha j} c_{\alpha k} f_\alpha^{eq} = h\frac{c^2}{3}\left(\delta_{ik} u_j + \delta_{jk} u_i + \delta_{ij} u_k\right) \tag{3.1.10}$$

Equation (3.1.6) can be also derived as a Taylor expansion of the Maxwell-Boltzmann distribution up to second order in the Mach number (Chen and Doolen 1998) or as a Taylor expansion up to second order in Mach number around the kinetic states that minimize an H function (Karlin et al. 1999).

The viscosity, ν, in the LBM for shallow water is recovered as follows

$$\nu = \frac{c^2}{3}\Delta t\left(\tau - \frac{1}{2}\right) \tag{3.1.11}$$

For the monolayer shallow water equations, the forcing term in the LBM is as follows

$$F_i = -gh\frac{\partial z_b}{\partial x_i} + C_b u_i \sqrt{u_i u_i} + \frac{\rho_a}{\rho_w} C_W u_{Wi} \sqrt{u_{Wi} u_{Wi}} + F_{Ci} \tag{3.1.12}$$

3.2 Recovery of Shallow Water Equations in D2Q9

To ensure that the LBGK model solves the shallow water equations with proper LB parameters, the moments in equations (3.1.7) – (3.1.10) are used to show the recovery of the shallow water equations (2.1.1) – (2.1.2).up to second order by Chapman-Enskog multi-scale analysis. Similar recovery work for single-layer shallow water equations can be found in (Zhou 2004). To recover the multi-layer shallow water equations without the forcing term, $f_\alpha\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t\right)$ is expanded around $\left(\mathbf{x}, t\right)$ using the Taylor series expansion:

$$f_\alpha(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t) = f_\alpha(\mathbf{x}, t) + \sum_{n=1}^{\infty} \frac{d^n f_\alpha(\mathbf{x}, t)}{dt^n} \frac{\Delta t^n}{n!} \tag{3.2.1}$$

where $d_t = \partial_t + \mathbf{c}_i \cdot \nabla$ is the total derivative with respect to time. For multi-scale analysis, the Chapman-Enskog expansion is adopted as follows:

$$\partial_i = \varepsilon \partial_{1i} \tag{3.2.2}$$

$$\partial_t = \varepsilon \partial_{t1} + \varepsilon^2 \partial_{t2} \tag{3.2.3}$$

Furthermore, the particle distribution functions, f_α, are perturbed around f_α^{eq} in terms of Knudsen number ε (Takashi 1997):

$$f_\alpha = f_\alpha^{eq} + \varepsilon f_\alpha^{(1)} + \varepsilon^2 f_\alpha^{(2)} + \varepsilon^3 f_\alpha^{(3)} + O(\varepsilon^4) \tag{3.2.4}$$

where $f_\alpha^{(k)}$ are the perturbation terms. Introducing equations (3.2.1)-(3.2.4) into equation (3.1.3) and grouping terms of the same order in ε, the differential equations up to second order are

$$O(\varepsilon): \quad \Delta t \left\{ \frac{\partial}{\partial t_1} + c_{\alpha i} \frac{\partial}{\partial x_{1i}} \right\} f_\alpha^{eq} = -\frac{1}{\tau} f_\alpha^{(1)} \tag{3.2.5}$$

$$O(\varepsilon^2):\quad \Delta t\left(\frac{\partial}{\partial t_2}f_\alpha^{eq}+\left\{\frac{\partial}{\partial t_1}+c_{\alpha i}\frac{\partial}{\partial x_{1i}}\right\}f_\alpha^{(1)}\right)+\frac{\Delta t^2}{2}\left(\frac{\partial}{\partial t_1}+c_{\alpha i}\frac{\partial}{\partial x_{1i}}\right)^2 f_\alpha^{eq}=-\frac{1}{\tau}f_\alpha^{(2)} \tag{3.2.6}$$

Taking $\sum_\alpha$Eq.(3.2.5), $\sum_\alpha e_{\alpha i}$Eq.(3.2.5), $\sum_\alpha$Eq.(3.2.6), and $\sum_\alpha e_{\alpha i}$Eq.(3.2.6), one can obtain

$$\begin{aligned} &O(\varepsilon):\frac{\partial M_0}{\partial t}+\frac{\partial M_{1i}}{\partial x_i}=0\\ &\frac{\partial M_{1i}}{\partial t}+\frac{\partial M_{2ij}}{\partial x_j}=0 \end{aligned} \tag{3.2.7}$$

$$\begin{aligned} &O(\varepsilon^2):\frac{\partial M_0}{\partial t_2}=0\\ &\frac{\partial M_{1i}}{\partial t_2}=\frac{\partial}{\partial x_{1j}}\left[\Delta t\left(\tau-\frac{1}{2}\right)\left(\frac{\partial M_{2ij}}{\partial t_1}+\frac{\partial M_{3ijk}}{\partial x_{1k}}\right)\right] \end{aligned} \tag{3.2.8}$$

where

$$\begin{aligned} &M_0=\sum_\alpha f_\alpha,\quad M_{1i}=\sum_\alpha f_\alpha c_{\alpha i}\\ &M_{2ij}=\sum_\alpha f_\alpha c_{\alpha i}c_{\alpha j},\quad M_{3ijk}=\sum_\alpha f_\alpha c_{\alpha i}c_{\alpha j}c_{\alpha k} \end{aligned} \tag{3.2.9}$$

Introducing equation (3.2.9) into equations (3.2.7) and (3.2.8), the differential equations for the first and second order terms become

$$\begin{aligned} &O(\varepsilon):\frac{\partial h}{\partial t_1}+\frac{\partial h\,u_i}{\partial x_{1i}}=0\\ &\frac{\partial h\,u_i}{\partial t_1}+\frac{\partial h\,u_i u_j}{\partial x_{1j}}=-\frac{\partial}{\partial x_{1i}}\left(h\,c_s^2\right) \end{aligned} \tag{3.2.10}$$

$$\begin{aligned} &O(\varepsilon^2):\frac{\partial h}{\partial t_2}=0\\ &\frac{\partial h\,u_i}{\partial t_2}=\frac{\partial}{\partial x_{1j}}\left[\Delta t\left(\tau-\frac{1}{2}\right)\frac{\partial}{\partial t_1}\left(\delta_{ij}h\,c_s^2+h\,u_i u_j\right)\right]\\ &+\frac{\partial}{\partial x_{1j}}\left[\Delta t\left(\tau-\frac{1}{2}\right)\frac{\partial}{\partial x_{1k}}\left(\frac{c^2}{3}\left(\delta_{jk}h\,u_i+\delta_{ik}h\,u_j+\delta_{ij}h\,u_k\right)\right)\right] \end{aligned} \tag{3.2.11}$$

Selecting the relaxation time

$$\tau = 0.5 + 3\frac{\nu}{\Delta t c^2} = 0.5 + 3\frac{\Delta t \nu}{\Delta x^2} \tag{3.2.12}$$

and the squared speed of sound

$$c_s^2 = \frac{1}{2} gh \tag{3.2.13}$$

Eequations (3.2.10) and (3.2.11) become

$$\begin{aligned} &O(\varepsilon): \frac{\partial h}{\partial t_1} + \frac{\partial h\ u_i}{\partial x_{1i}} = 0 \\ &\frac{\partial h\ u_i}{\partial t_1} + \frac{\partial h\ u_i u_j}{\partial x_{1j}} = -\frac{\partial}{\partial x_{1i}}\left(\frac{gh^2}{2}\right) \end{aligned} \tag{3.2.14}$$

$$\begin{aligned} &O(\varepsilon^2): \frac{\partial h}{\partial t_2} = 0 \\ &\frac{\partial h\ u_i}{\partial t_2} = \nu \frac{\partial^2 h\ u_i}{\partial x_{1j} \partial x_{1j}} + \underbrace{\nu \frac{3}{e^2} \frac{\partial}{\partial t_1}\left[\frac{\partial}{\partial x_{1i}}\left(\frac{gh^2}{2}\right) + \frac{\partial h\ u_i u_j}{\partial x_{1j}}\right]}_{\mathbf{I}} \end{aligned} \tag{3.2.15}$$

The term $\mathbf{I}$ represents the numerical error. Substituting the second equation of (3.2.14) into the term $\mathbf{I}$, it becomes the second derivative of $h\ u_i$ with respect to t_1, which is small compared to the first derivative and is neglected. Combining the first order terms, equation (3.2.14), and second order terms, equations (3.2.15), the shallow water equations are recovered:

$$\begin{aligned} &\frac{\partial h}{\partial t} + \frac{\partial h\ u_i}{\partial x_i} = 0 \\ &\frac{\partial h\ u_i}{\partial t} + \frac{\partial h\ u_i u_j}{\partial x_j} + \frac{\partial}{\partial x_i}\left(\frac{gh^2}{2}\right) = \nu \frac{\partial^2 h\ u_i}{\partial x_j \partial x_j} \end{aligned} \tag{3.2.16}$$

3.3 LBM with MRT Collision Operator

This study introduces a lattice Boltzmann model to solve the shallow water equations based on the generalized lattice Boltzmann equation (GLBE) with the multiple-relaxation-time

(MRT) collision operator (d'Humieres et al. 2002; Ginzburg 2007; Guo et al. 2008). The evolution equation using the MRT lattice Boltzmann equation is

$$\mathbf{f}\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t\right)-\mathbf{f}\left(\mathbf{x},t\right)=-\mathrm{M}^{-1}\mathrm{S}\left[\mathbf{m}\left(\mathbf{x},t\right)-\mathbf{m}^{eq}\left(\mathbf{x},t\right)\right]+\frac{\Delta t}{6c^2}\mathbf{F}_\alpha, \tag{3.3.1}$$

where $\mathbf{f}=\{f_\alpha,\alpha=0,1,2,\cdots,8\}$ is a nine-dimensional column vector of particle distribution functions for D2Q9 lattice. $\mathbf{m}$ and $\mathbf{m}^{eq}$ are nine-dimensional column vectors of moments and their equilibria, respectively. M is a 9×9 dimensional transformation matrix that transforms the particle distribution functions and equilibrium distribution functions (EDFs) from velocity space to moment space, which makes $\mathbf{m}=\mathrm{M}\mathbf{f}$ and $\mathbf{m}^{eq}=\mathrm{M}\mathbf{f}^{eq}$. $\mathbf{f}^{eq}=\{f_\alpha^{eq},\alpha=0,1,2,\cdots,8\}$ is a nine-dimensional column vector of the EDFs. $\mathrm{S}=\mathrm{diag}\left(s_0,s_1,\cdots,s_8\right)$ is a 9×9 diagonal matrix, where $s_\alpha\geq0$ are the relaxation rates. $\mathbf{F}_\alpha=\{\sum_i c_{\alpha i}F_i,\alpha=0,1,2,\cdots,8\}$ is an external force along the α direction.

Equation (3.3.1) is the evolution equation for the particle distribution functions. The left hand side represents particle transport by pure advection executed in the streaming velocity space; and the right hand side represents the collision process modeled by linear relaxation processes executed in the moment space. The lattice Boltzmann algorithm consists of two steps: streaming and collision. In each time step, the particle distribution functions arrive to their neighboring nodes at the same time through prescribed lattice connections. The forcing terms are given by

$$F_i=F_i\left(\mathbf{x}+\frac{1}{2}\mathbf{c}_\alpha\Delta t,t\right)=\left(-gh\frac{\partial z_b}{\partial x_i}-gn_b^2\frac{u_i\sqrt{u_ju_j}}{h^{1/3}}\right)\Bigg|_{\mathbf{x}+\frac{1}{2}\mathbf{c}_\alpha\Delta t,t}. \tag{3.3.2}$$

The transformation matrix M in the GLBE is constructed such that MM^{T} is a diagonal matrix, where the column vectors $\{\mathbf{b}_\alpha\}$ of M^{T} are mutually orthogonal (Lallemand and Luo

2000; d'Humieres et al. 2002; Lallemand and Luo 2003). The transformation matrix M for D2Q9 is given by Lallemand and Luo (Lallemand and Luo 2000):

$$\mathrm{M}=\begin{pmatrix}\mathbf{b}_0^{\mathrm{T}}\\ \mathbf{b}_1^{\mathrm{T}}\\ \mathbf{b}_2^{\mathrm{T}}\\ \mathbf{b}_3^{\mathrm{T}}\\ \mathbf{b}_4^{\mathrm{T}}\\ \mathbf{b}_5^{\mathrm{T}}\\ \mathbf{b}_6^{\mathrm{T}}\\ \mathbf{b}_7^{\mathrm{T}}\\ \mathbf{b}_8^{\mathrm{T}}\end{pmatrix}=\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2\\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1\\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1\\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1\\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1\\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1\\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1\end{pmatrix}. \tag{3.3.3}$$

Inserting matrices M and S into equation (3.3.1), the evolution equation in the direction α becomes [37]

$$f_\alpha\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t,t+\Delta t\right)-f_\alpha\left(\mathbf{x},t\right)=-\sum_{\beta=0}^{8}\frac{s_\beta \mathrm{b}_{\beta\alpha}}{\left\|\mathbf{b}_\beta\right\|}\left[m_\beta\left(\mathbf{x},t\right)-m_\beta^{\,eq}\left(\mathbf{x},t\right)\right]+\frac{\Delta t}{6c^2}\sum_i c_{\alpha i}F_i\,. \tag{3.3.4}$$

The moments for the GLBE applied to the shallow water equations are (Li and Huang 2008)

$$\mathbf{m}=\left(h,e,\varepsilon,j_x,q_x,j_y,q_y,p_{xx},p_{xy}\right), \tag{3.3.5}$$

where $m_0=h$ is the water depth, $m_1=e$ is related to the total energy, $m_2=\varepsilon$ is related to the energy square, $\left(m_3,m_5\right)=\left(j_x,j_y\right)=\left(hu_x,hu_y\right)$ are the flow momenta, $\left(m_4,m_6\right)=\left(q_x,q_y\right)$ are related to the head flux, and $m_7=p_{xx}$ and $m_8=p_{xy}$ are related to the diagonal and off-diagonal components of the stress tensor, respectively. In the application to the shallow water equations, the conserved moments are the water depth and the flow momenta:

$$h=\sum\nolimits_\alpha f_\alpha\,, \tag{3.3.6}$$

$$hu_i=\sum\nolimits_\alpha c_{\alpha i}f_\alpha\,. \tag{3.3.7}$$

The remaining moments are not conserved quantities. Using equations (4.1.4) and (3.3.3), the equilibrium moments, $\mathbf{m}^{eq} = \mathrm{M}\mathbf{f}^{eq}$, are

$$m_1^{eq} = -4h + \frac{3gh^2}{c^2} + \frac{3h\left(u_x^2 + u_y^2\right)}{c^2}, \quad m_2^{eq} = 4h - \frac{9gh^2}{2c^2} - \frac{3h\left(u_x^2 + u_y^2\right)}{c^2}, \tag{3.3.8}$$

$$m_4^{eq} = -\frac{hu_x}{c}, \quad m_6^{eq} = -\frac{hu_y}{c}, \tag{3.3.9}$$

$$m_7^{eq} = \frac{3h\left(u_x^2 - u_y^2\right)}{c^2}, \quad m_8^{eq} = \frac{hu_x u_y}{c^2}. \tag{3.3.10}$$

The equilibria of the conserved moments $\left(m_0, m_3, \text{and } m_5\right)$ are equal to themselves. Therefore, the relaxation rates, s_0, s_3, and s_5, have no effect on LBM solutions. With the moment equilibria given by equations (3.3.8)-(3.3.10), the shallow water equations can be recovered with the shear and bulk viscosities given by Lallemand and Luo (Lallemand and Luo 2000)

$$v = \frac{1}{3}\left(\frac{1}{\mathrm{s}_7} - \frac{1}{2}\right)c\Delta x = \frac{1}{3}\left(\frac{1}{\mathrm{s}_8} - \frac{1}{2}\right)c\Delta x, \tag{3.3.11}$$

$$\zeta = \frac{1}{6}\left(\frac{1}{\mathrm{s}_1} - \frac{1}{2}\right)c\Delta x. \tag{3.3.12}$$

Setting $\mathrm{s}_\alpha = 1/\tau$, the GLBE, equation (3.3.1), reduces to the single relaxation time (SRT) LBM model, referred to the LBGK model, for shallow water equations (Salmon 1999; Zhou 2002). For the LBGK model in D2Q9, the shear and bulk viscosities become

$$v = 2\zeta = \frac{1}{3}\left(\tau - \frac{1}{2}\right)c\Delta x. \tag{3.3.13}$$

The SRT constrains the non-conserved moments (free parameters) and makes the LBGK particularly prone to numerical instabilities. Insufficient dissipation from non-conserved moments is unable to eliminate the fluctuations in water depth when τ is close to $1/2$, which is

needed for very small kinematic viscosity of $v = 1\times10^{-6}\ m^2/s$ used in this study. One way to overcome the stability problem is either to use very fine grids or large viscosity values in the SRT-LBM model. The MRT-LBM model has no such a problem because the relaxation rates corresponding to non-conserved moments can be selected to attain stable solutions.

3.4 LBM for Anisotropic Advection Dispersion

The two-relaxation-time (TRT) collision operator (Ginzburg 2005; Servan-Camas and Tsai 2008; Servan-Camas and Tsai 2009) is sufficient for solute transport in shallow water flow. The TRT collision operator is a particular form of the MRT collision operator with two unique relaxation rates where $s_1 = s_2 = s_7 = s_8 = 1/\tau_a$ and $s_4 = s_6 = 1/\tau_s$ in equation (3.3.1). The TRT collision operator does not have any macroscopic advantage over the MRT collision operator for the mass transport equation, but does have a computational advantage based on efficiency and simplicity of analysis and coding (Ginzburg 2007). The TRT-LBM model is

$$\begin{aligned} f_\alpha\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t\right) - f_\alpha\left(\mathbf{x},t\right) = &-\frac{1}{\tau_s}\left(f_\alpha^s\left(\mathbf{x},t\right) - f_\alpha^{seq}\left(\mathbf{x},t\right)\right) \\ &-\frac{1}{\tau_a}\left(f_\alpha^a\left(\mathbf{x},t\right) - f_\alpha^{aeq}\left(\mathbf{x},t\right)\right) + \Delta t S_{C\alpha}\left(\mathbf{x},t\right) \end{aligned}, \tag{3.4.1}$$

where f_α^s and f_α^{seq} are the symmetric parts of the particle distribution function and equilibrium distribution function, respectively; f_α^a and f_α^{aeq} are the anti-symmetric parts of the particle distribution function and equilibrium distribution function, respectively; τ_s and τ_a are the symmetric relaxation time and anti-symmetric relaxation time, respectively. $S_{C\alpha}$ is an external force along the α direction. Equation (3.4.1) reduces to the SRT-LBM model when $\tau_s = \tau_a$. For the TRT, the symmetric relaxation time was suggested by Servan-Camas and Tsai (Servan-Camas and Tsai 2009):

$$\tau_s = \frac{1}{2} + \frac{1}{12(\tau_a - 0.5)}. \tag{3.4.2}$$

In the TRT collision operator, particle distribution functions are relaxed to the equilibrium state by relaxing their symmetric and anti-symmetric parts separately, which are given by

$$\begin{aligned} f_\alpha^s &= \frac{f_\alpha + \overline{f_\alpha}}{2}; \qquad f_\alpha^a = \frac{f_\alpha - \overline{f_\alpha}}{2} \\ f_\alpha^{seq} &= \frac{f_\alpha^{eq} + \overline{f_\alpha^{eq}}}{2}; \qquad f_\alpha^{aeq} = \frac{f_\alpha^{eq} - \overline{f_\alpha^{eq}}}{2} \end{aligned}, \tag{3.4.3}$$

where $\overline{g_\alpha}$ and $\overline{g_\alpha^{eq}}$ are the distribution functions and EDF along opposite direction of α, respectively. The zeroth, first, and second moments of the EDFs are

$$\sum_\alpha f_\alpha^{eq} = \sum_\alpha f_\alpha^{seq} = hC, \qquad \sum_\alpha f_\alpha^{aeq} = 0, \tag{3.4.4}$$

$$\sum_\alpha c_{\alpha i} f_\alpha^{eq} = \sum_\alpha c_{\alpha i} f_\alpha^{aeq} = hCu_i, \qquad \sum_\alpha c_{\alpha i} f_\alpha^{seq} = 0, \tag{3.4.5}$$

$$\sum_\alpha c_{\alpha i} c_{\alpha j} f_\alpha^{eq} = \sum_\alpha c_{\alpha i} c_{\alpha j} f_\alpha^{seq} = C\left(\frac{hD_{ij}}{\Delta t(\tau_a - 1/2)} + hu_i u_j \right), \qquad \sum_\alpha c_{\alpha i} c_{\alpha j} f_\alpha^{aeq} = 0. \tag{3.4.6}$$

The EDFs applied to the AADE are (Servan-Camas and Tsai 2010)

$$\begin{aligned} f_\alpha^{eq} &= C\omega_\alpha \left((2\tau_a - 1)\frac{\hat{c}_{s\alpha}^2}{c^2} + (2 - 2\tau_a)\frac{c_{s\alpha}^2}{c^2} + \frac{h\mathbf{u}\cdot\mathbf{c}_\alpha}{c^2} + \frac{3}{2}\frac{h(\mathbf{u}\cdot\mathbf{c}_\alpha)^2}{c^4} - \frac{1}{2}\frac{h\mathbf{u}\cdot\mathbf{u}}{c^2} \right) \quad , \alpha > 0 \\ f_\alpha^{eq} &= hC - \sum_{\alpha=1}^{8} f_\alpha^{eq} \end{aligned}, \tag{3.4.7}$$

where $c_{s\alpha}$ is the multispeed of sound (MSS), a numerical parameter related to the coefficients of anisotropy. The MSS for the eight directions are given as follows:

$$
\begin{aligned}
c_{s1}^2 &= c_{s3}^2 = \frac{3c_{sxx}^2 - c_{sxx}c_{syy}}{2} \\
c_{s2}^2 &= c_{s4}^2 = \frac{3c_{syy}^2 - c_{sxx}c_{syy}}{2} \\
c_{s5}^2 &= c_{s7}^2 = 3c_{sxy}^2 + c_{sxx}c_{syy} \\
c_{s6}^2 &= c_{s8}^2 = -3c_{sxy}^2 + c_{sxx}c_{syy}
\end{aligned}
\quad , \tag{3.4.8}
$$

where c_{sij}^2 is the anisotropic local squared speed of sound in each lattice grid point. From the recovery procedure, one can derive c_{sij}^2 relating to the product of the dispersion tensor and the water depth:

$$
c_{sij}^2(\mathbf{x},t) = \frac{h(\mathbf{x},t)\,D_{ij}(\mathbf{x},t)}{\Delta t(\tau_a - 1/2)}. \tag{3.4.9}
$$

$\hat{c}_{s\alpha}^2(\mathbf{x},t)$ in equation (3.4.7) is the squared directional speed of sound (DSS) to account for heterogeneity. $\hat{c}_{s\alpha}^2(\mathbf{x},t)$ is calculated as the arithmetic mean of the MSS along the α direction across adjacent lattice grids:

$$
\hat{c}_{s\alpha}^2(\mathbf{x},t) = \frac{c_{s\alpha}^2(\mathbf{x},t) + c_{s\alpha}^2(\mathbf{x}+\mathbf{c}_\alpha\Delta t,t)}{2}. \tag{3.4.10}
$$

The forcing terms are given by

$$
\begin{aligned}
S_{C0}(\mathbf{x},t) &= S_C(\mathbf{x},t) + 0.5\left[S_C(\mathbf{x},t) - S_C(\mathbf{x},t-\Delta t)\right] \\
S_{C\alpha}(\mathbf{x},t) &= 0 \qquad \alpha > 0
\end{aligned}
. \tag{3.4.11}
$$

To ensure that the TRT-LBM model solves the AADE in shallow water with proper LB parameters, the recovery of the AADE, equation (2.3.1), up to second order by Chapman-Enskog multi-scale analysis utilizing the moments in equations (3.4.4)-(3.4.6) can be found in Servan-Camas and Tsai (Servan-Camas and Tsai 2009; Servan-Camas and Tsai 2010).

3.5 Boundary and Initial Conditions

3.5.1 Introduction

In order to simulate shallow water flow problems, suitable boundary and initial conditions must be provided. Boundary and initial conditions in the lattice Boltzmann formulation rely on connecting the macroscopic boundary conditions in the physical problem to mesoscopic boundary conditions on the distributions functions, f_α. Consider the idealized domain shown in Figure 3.2.

3.5.2 Periodic Boundary Conditions

In some cases, periodic boundary conditions may be necessary. One such case is when a flow region consists of a number of same sub regions where the flow pattern repeats itself. In this case, only one sub region is actually required to be modeled using a periodic boundary condition. Implementing periodic boundary conditions in the lattice Boltzmann formulation is achieved by setting the unknown distribution functions, f_1, f_5 and f_8 at the inflow boundary (see Figure 3.2) to the corresponding known distribution functions at the outflow boundary,

$$f_\alpha\left(i=1, j, t\right) = f_\alpha\left(i=N_x, j, t\right), \qquad \alpha = 1,5,8 \tag{3.4.12}$$

and the unknown distribution functions, f_3, f_6 and f_7 at the inflow boundary,

$$f_\alpha\left(i=N_x, j, t\right) = f_\alpha\left(i=1, j, t\right), \qquad \alpha = 3,6,7 \tag{3.4.13}$$

Similarly, a periodic boundary condition in the y direction can be formulated.

3.5.3 Solid Boundary Conditions

Boundary conditions for solid boundaries such as impermeable boundaries or structures in the flow region are prescribed by applying no-slip or free-slip at these boundaries to prescribe zero velocity or zero normal velocity at the boundary, respectively. Implementing no-slip and free-slip boundary conditions in the lattice Boltzmann formulation is simple using bounce-back

scheme (Chen and Doolen 1998). The basic idea behind the bounce-back scheme is that unknown distribution functions are a function of the known distribution functions incident on the boundary, defined by symmetry conditions.

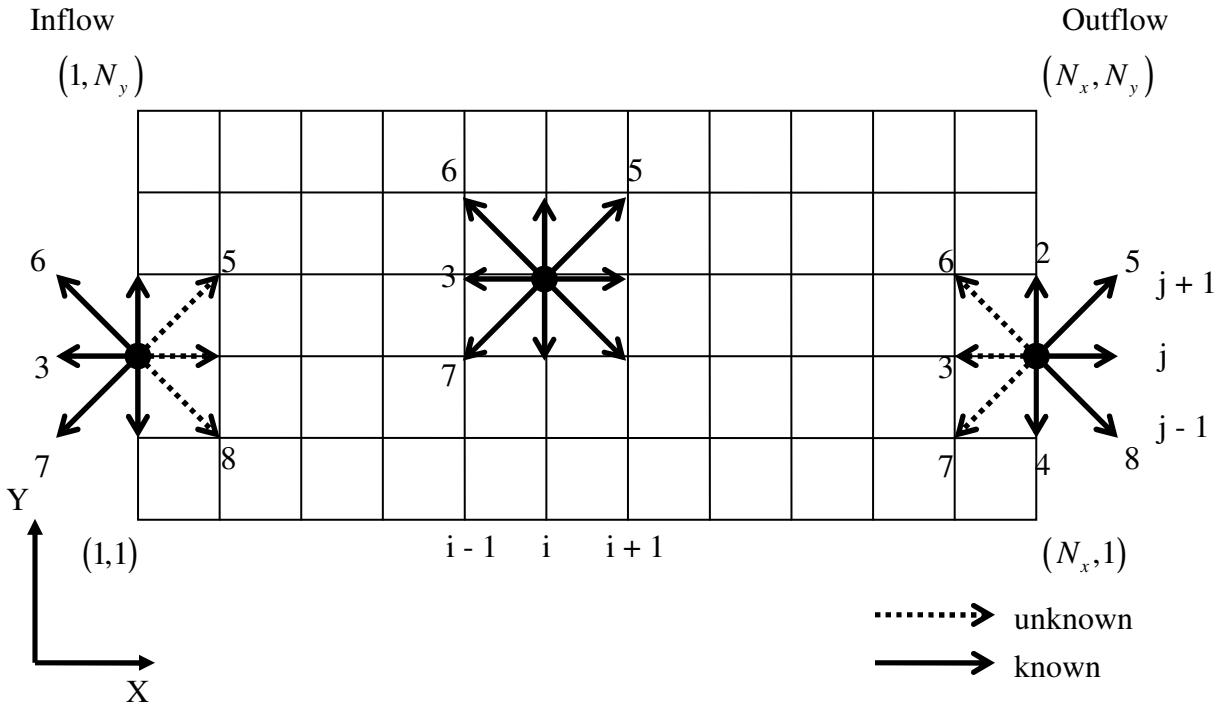


Figure 3.2 Definition sketch lattice nodes for inflow, outflow and south solid boundaries.

A no-slip boundary condition is achieved by setting the unknown distribution functions, f_2, f_5 and f_6 at the south boundary (see Figure 3.3) to the known distributions, f_4, f_7 and f_8 corresponding to the opposite directions,

$$f_2 = f_4, \qquad f_5 = f_7, \qquad f_6 = f_8. \tag{3.4.14}$$

This ensures a zero flux across the boundary in both the normal and tangential directions. A free-slip boundary condition is achieved in a similar way; however, it results in a zero flux across the normal direction and non-zero flux along the tangential direction. This is achieved by setting the unknown distribution functions, f_2, f_5 and f_6 at the south boundary to the known distributions,

f_4, f_7 and f_8 corresponding to the opposite direction along the normal direction and reflected direction along the tangential direction,

$$f_2 = f_4, \qquad f_5 = f_8, \qquad f_6 = f_7. \tag{3.4.15}$$

Following the same bounce back scheme, no-slip and free-slip boundary conditions can be implemented on the east and west boundaries.

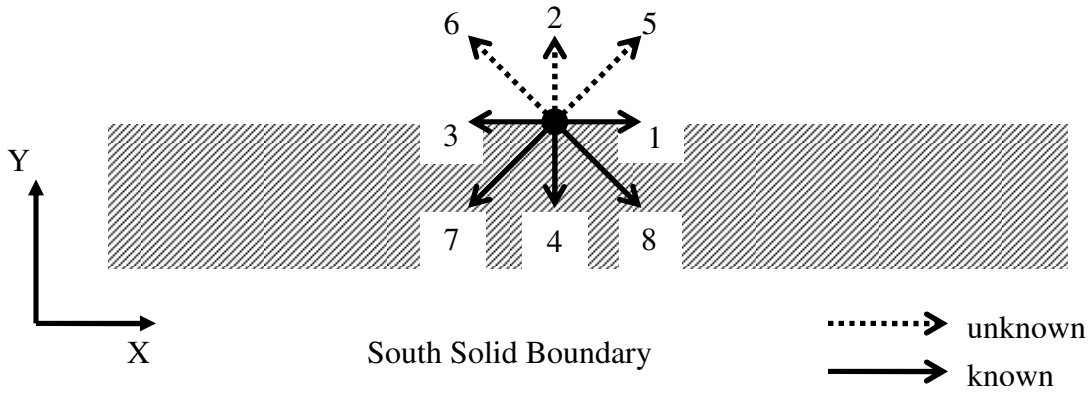


Figure 3.3 Definition sketch lattice nodes the south solid boundaries.

3.5.4 Open Boundary Conditions

Boundary conditions for open boundaries such as inflow, outflow and seaward boundary conditions are prescribed by giving the macroscopic boundary values or functions at the boundary, i.e., constant water depth, water depth defined by tidal function, discharge, etc. If the velocity and depth are known, the unknown distribution functions, f_α, can be computed using Equations (3.1.4) and (3.1.5) following the method described by Zou and He (1997). At the inflow boundary (see Figure 3.2), Equations (3.1.4) and (3.1.5) lead to three equations,

$$f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8 = h \tag{3.4.16}$$

$$c\left(f_1 + f_5 + f_8\right) - c\left(f_3 + f_6 + f_7\right) = hu_x \tag{3.4.17}$$

$$c\left(f_2 + f_5 + f_6\right) - c\left(f_4 + f_7 + f_8\right) = hu_y \tag{3.4.18}$$

If $u_y = 0$ is assumed, solving the above three equations for the unknown distribution functions, f_1, f_5 and f_8 results in

$$f_1 = f_3 + \frac{2hu_x}{3c} \tag{3.4.19}$$

$$f_5 = \frac{hu_x}{6c} + f_7 + \frac{f_4 - f_2}{2} \tag{3.4.20}$$

$$f_8 = \frac{hu_x}{6c} + f_6 + \frac{f_2 - f_4}{2} \tag{3.4.21}$$

Following the same procedure, the unknown distribution functions, f_3, f_6 and f_7 for the outflow boundary can be determined using

$$f_3 = f_1 - \frac{2hu_x}{3c} \tag{3.4.22}$$

$$f_6 = -\frac{hu_x}{6c} + f_8 + \frac{f_2 - f_4}{2} \tag{3.4.23}$$

$$f_7 = -\frac{hu_x}{6c} + f_5 + \frac{f_4 - f_2}{2} \tag{3.4.24}$$

3.5.5 Initial Conditions

The initial conditions for a physical problem to be modeled are given in form of macroscopic variables which is normal practice in traditional numerical methods. Since the lattice Boltzmann formulation is based on solving Equation (3.1.3), the initial conditions must be written in terms of the distribution function f_α. Given the initial macroscopic boundary conditions, h, u_x, and u_y, the EDF, f_α^{eq}, is computed and used as initial conditions for f_α, i.e. $f_\alpha = f_\alpha^{eq}\left(h(\mathbf{x}, t=0), u_x(\mathbf{x}, t=0), u_y(\mathbf{x}, t=0)\right)$.

4 LBM FOR MULTI-LAYER SHALLOW WATER EQUATIONS

4.1 MRT Collision Operator

The LB model is used to solve the shallow water equations for each layer. The LB equation for describing dynamics of local particle distribution functions in a discrete velocity field is (Chen and Doolen 1998; Zhou 2004):

$$f_\alpha^{(\ell)}\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t, t+\Delta t\right) = f_\alpha^{(\ell)}\left(\mathbf{x},t\right) - \sum_{\beta=0}^{8}\frac{s_\beta \mathrm{b}_{\beta\alpha}}{\left\|\mathbf{b}_\beta\right\|}\left[m^{(\ell)n}{}_\beta\left(\mathbf{x},t\right) - m_\beta^{((\ell)eq)n}\left(\mathbf{x},t\right)\right] + \frac{\Delta t c_{\alpha i}}{6e^2}F_i^{(\ell)}\left(\mathbf{x},t\right) \quad (4.1.1)$$

where $f_\alpha^{(\ell)}$ is the particle distribution function moving along the α direction for the layer ℓ, $f_\alpha^{(\ell)eq}$ is the equilibrium distribution function (EDF) along the α direction for the layer ℓ. The second term on the right hand side of equation (4.1.1) is the MRT collision collision operator, which represents changes in $f_\alpha^{(\ell)}$ due to particle collisions. F_i is the external force per unit mass in the shallow water equations (Zhou 2004). The external forces are defined in macroscopic variables and their contribution distributed along the α directions. Equation (4.1.1) consists of a streaming step and a collision step (Succi 2001; Zhou 2004). Based on equation (4.1.1), a multi-layer LBGK model is constructed.

The local water height and flow velocity for each layer are calculated as the zeroth and first moments of the distribution functions:

$$h^{(\ell)} = \sum\nolimits_\alpha f_\alpha^{(\ell)} \quad (4.1.2)$$

$$h^{(\ell)}u_i^{(\ell)} = \sum\nolimits_\alpha c_{\alpha i} f_\alpha^{(\ell)} \quad (4.1.3)$$

The EDFs (Chen and Doolen 1998) applied to the multi-layer shallow water equations are

$$f_\alpha^{(\ell)eq} = h^{(\ell)}\omega_\alpha\left(\frac{c_s^2}{c^2}+\frac{\mathbf{u}^{(\ell)}\cdot\mathbf{c}_\alpha}{c^2}+\frac{3}{2}\frac{\left(\mathbf{u}^{(\ell)}\cdot\mathbf{c}_\alpha\right)^2}{c^4}-\frac{1}{2}\frac{\mathbf{u}^{(\ell)}\cdot\mathbf{u}^{(\ell)}}{c^2}\right) \quad ,\alpha>0$$
$$f_0^{(\ell)eq} = h^{(\ell)} - \sum_{\alpha=1}^{8} f_\alpha^{(\ell)eq} \tag{4.1.4}$$

where c_s^2 is known as the squared speed of sound, which in LBM for shallow water equations is a numerical parameter that relates to the wave celerity, $C=\sqrt{gH}$, i.e., $c_s^2 = C^2/2 = gH/2$. It is noted that the weighting coefficients ω_α remain the same as in monolayer LBGK formulation since each layer has the same planar discretization. The EDFs for the multi-layer LBGK have the same form as the monolayer and follow the same constraints on the zeroth, first, second, and third moments for each layer:

$$\sum\nolimits_\alpha f_\alpha^{(\ell)eq} = h^{(\ell)} \tag{4.1.5}$$

$$\sum\nolimits_\alpha c_{\alpha i} f_\alpha^{(\ell)eq} = h^{(\ell)}u_i^{(\ell)} \tag{4.1.6}$$

$$\sum\nolimits_\alpha c_{\alpha i}c_{\alpha j} f_\alpha^{(\ell)eq} = h^{(\ell)}\left(\delta_{ij}c_s^2 + u_i^{(\ell)}u_j^{(\ell)}\right) \tag{4.1.7}$$

$$\sum\nolimits_\alpha c_{\alpha i}c_{\alpha j}c_{\alpha k} f_\alpha^{(\ell)eq} = h^{(\ell)}\frac{c^2}{3}\left(\delta_{ik}u_j^{(\ell)} + \delta_{jk}u_i^{(\ell)} + \delta_{ij}u_k^{(\ell)}\right) \tag{4.1.8}$$

The forcing term is given as

$$\begin{aligned} F_i^{(l)} &= F_{Wi}^{(\ell)} + F_{Pi}^{(\ell)} + F_{NCi}^{(\ell)} + F_{Ci}^{(\ell)} + F_\rho^{(\ell)} \\ &\quad -\kappa\delta_{1l}u_i^{(\ell)} + 2\mu\left(1-\delta_{Ml}\right)\frac{u_i^{(\ell+1)}-u_i^{(\ell)}}{h_i^{(\ell+1)}-h_i^{(\ell)}} - 2\mu\left(1-\delta_{1l}\right)\frac{u_i^{(\ell)}-u_i^{(\ell-1)}}{h_i^{(\ell)}-h_i^{(\ell-1)}} \end{aligned} \tag{4.1.9}$$

The bed friction is now considered in the kappa term and the vertical viscosity forces in the mu term.

4.2 Recovery of Multi-layer Shallow Water Equations

This section presents the recovery of the multi-layer shallow water equation by multi-scale analysis. Without loss of generality, the multi layer recovery begins with results of multi-scale analysis of the LBM for shallow water equations, i.e., equations (3.2.10), (3.2.11) and (3.2.13), applied to layer ℓ of the multi-layer LBM.

$$\begin{aligned} &O(\varepsilon): \frac{\partial h^{(\ell)}}{\partial t_1} + \frac{\partial h^{(\ell)} u_i^{(\ell)}}{\partial x_{1i}} = 0 \\ &\frac{\partial h^{(\ell)} u_i^{(\ell)}}{\partial t_1} + \frac{\partial h^{(\ell)} u_i^{(\ell)} u_j^{(\ell)}}{\partial x_{1j}} = -\frac{\partial}{\partial x_{1i}}\left(h^{(\ell)} c_s^2\right) \end{aligned} \tag{4.1.10}$$

$$\begin{aligned} &O(\varepsilon^2): \frac{\partial h^{(\ell)}}{\partial t_2} = 0 \\ &\frac{\partial h^{(\ell)} u_i^{(\ell)}}{\partial t_2} = \frac{\partial}{\partial x_{1j}}\left[\Delta t\left(\tau - \frac{1}{2}\right)\frac{\partial}{\partial t_1}\left(\delta_{ij} h^{(\ell)} c_s^2 + h^{(\ell)} u_i^{(\ell)} u_j^{(\ell)}\right)\right] \\ &+ \frac{\partial}{\partial x_{1j}}\left[\Delta t\left(\tau - \frac{1}{2}\right)\frac{\partial}{\partial x_{1k}}\left(\frac{c^2}{3}\left(\delta_{jk} h^{(\ell)} u_i^{(\ell)} + \delta_{ik} h^{(\ell)} u_j^{(\ell)} + \delta_{ij} h^{(\ell)} u_k^{(\ell)}\right)\right)\right] \end{aligned} \tag{4.1.11}$$

Selecting the relaxation time

$$\tau = 0.5 + 3\frac{\nu}{\Delta t c^2} = 0.5 + 3\frac{\Delta t \nu}{\Delta x^2} \tag{4.1.12}$$

and the squared speed of sound

$$c_s^2 = \frac{1}{2} g \sum_{m=1}^{M} h^{(m)} = \frac{1}{2} gH \tag{4.1.13}$$

equations (4.2.1) and (4.2.2) become

$$\begin{aligned} &O(\varepsilon): \frac{\partial h^{(\ell)}}{\partial t_1} + \frac{\partial h^{(\ell)} u_i^{(\ell)}}{\partial x_{1i}} = 0 \\ &\frac{\partial h^{(\ell)} u_i^{(\ell)}}{\partial t_1} + \frac{\partial h^{(\ell)} u_i^{(\ell)} u_j^{(\ell)}}{\partial x_{1j}} = -\frac{\partial}{\partial x_{1i}}\left(\frac{g h^{(\ell)} \sum_{m=1}^{M} h^{(m)}}{2}\right) \end{aligned} \tag{4.1.14}$$

$$O\left(\varepsilon^2\right):\frac{\partial h^{(\ell)}}{\partial t_2}=0$$

$$\frac{\partial h^{(\ell)}u_i^{(\ell)}}{\partial t_2}=\nu\frac{\partial^2 h^{(\ell)}u_i^{(\ell)}}{\partial x_{1j}\partial x_{1j}}+\underbrace{\nu\frac{3}{c^2}\frac{\partial}{\partial t_1}\left[\frac{\partial}{\partial x_{1i}}\left(\frac{gh^{(\ell)}\sum_{m=1}^{M}h^{(m)}}{2}\right)+\frac{\partial h^{(\ell)}u_i^{(\ell)}u_j^{(\ell)}}{\partial x_{1j}}\right]}_{\mathbf{I}} \tag{4.1.15}$$

The term $\mathbf{I}$ represents the numerical error. Substituting the second equation of (4.2.5) into the term $\mathbf{I}$, it becomes the second derivative of $h^{(\ell)}u_i^{(\ell)}$ with respect to t_1, which is small compared to the first derivative and is neglected. Combining the first and second order terms in equations (4.2.5) and (4.2.6), the multi-layer shallow water equations are recovered:

$$\begin{aligned}&\frac{\partial h^{(\ell)}}{\partial t}+\frac{\partial h^{(\ell)}u_i^{(\ell)}}{\partial x_i}=0\\&\frac{\partial h^{(\ell)}u_i^{(\ell)}}{\partial t}+\frac{\partial h^{(\ell)}u_i^{(\ell)}u_j^{(\ell)}}{\partial x_j}+\frac{\partial}{\partial x_i}\left(\frac{gh^{(\ell)}\sum_{m=1}^{M}h^{(m)}}{2}\right)=\nu\frac{\partial^2 h^{(\ell)}u_i^{(\ell)}}{\partial x_j\partial x_j}\end{aligned} \tag{4.1.16}$$

4.3 Boundary and Initial Conditions

4.3.1 Introduction

Concerning the boundary conditions for the multi-layer lattice Boltzmann formulation, treatments for the boundary conditions for the monolayer shallow water equations are referred to. Periodic boundary, solid boundary, and initial conditions are straightforward to extend to the multi-layer lattice Boltzmann formulation as they involve exactly the same techniques presented in sections 3.3.2, 3.3.3 and 3.3.5 repeated for each layer. Open boundary conditions, however, need more attention as they involve macroscopic variables defined by the multi-layer shallow water equations.

4.3.2 Multi-layer Open Boundary Conditions

Based on the known velocity and depth, the unknown distribution functions in the monolayer formulation, f_α, were calculated based on the conservation constraints on the zero

and first moments in section 3.3.4. The unknown distribution functions in the multi-layer formulation, $f_\alpha^{(\ell)}$ are calculated in a similar manner for each layer, however, the equivalent local water heights, $h^{(\ell)}$, and velocities $u^{(\ell)}$ must be determined. The local water height is prescribed as

$$h^{(\ell)} = \frac{H}{M}, \qquad \ell = 1,...,M \ . \tag{4.2.1}$$

For the velocity components, a constant velocity along the vertical direction prescribed as

$$u_i^{(\ell)} = U_i, \qquad \ell = 1,...,M \ . \tag{4.2.2}$$

Similar to section 3.3.4, $u_y^{(l)} = 0$ is assumed, resulting in the following equations for the unknown distribution functions, $f_1^{(\ell)}$, $f_5^{(\ell)}$ and $f_8^{(\ell)}$:

$$f_1^{(\ell)} = f_3^{(\ell)} + \frac{2h^{(\ell)}u_x^{(\ell)}}{3c} \tag{4.2.3}$$

$$f_5^{(\ell)} = \frac{h^{(\ell)}u_x^{(\ell)}}{6c} + f_7^{(\ell)} + \frac{f_4^{(\ell)} - f_2^{(\ell)}}{2} \tag{4.2.4}$$

$$f_8^{(\ell)} = \frac{h^{(\ell)}u_x^{(\ell)}}{6c} + f_6^{(\ell)} + \frac{f_2^{(\ell)} - f_4^{(\ell)}}{2} \tag{4.2.5}$$

Following the same procedure, the unknown distribution functions, $f_3^{(\ell)}$, $f_6^{(\ell)}$ and $f_7^{(\ell)}$ for the outflow boundary are determined using

$$f_3^{(\ell)} = f_1^{(\ell)} - \frac{2h^{(\ell)}u_x^{(\ell)}}{3c} \tag{4.2.6}$$

$$f_6^{(\ell)} = -\frac{h^{(\ell)}u_x^{(\ell)}}{6c} + f_8^{(\ell)} + \frac{f_2^{(\ell)} - f_4^{(\ell)}}{2} \tag{4.2.7}$$

$$f_7^{(\ell)} = -\frac{h^{(\ell)}u_x^{(\ell)}}{6c} + f_5^{(\ell)} + \frac{f_4^{(\ell)} - f_2^{(\ell)}}{2} \tag{4.2.8}$$

4.4 Multi-Layer LB Algorithm

With the solution known at time n, the solution at time $n+1$ is calculated with an explicit treatment of quantities local to a layer, i.e., the left hand side of equations (2.2.1) and(2.2.2), the kinematic viscosity term in equation(2.2.6), the wind-driven forcing term, the bed slope forcing term, the non-conservative pressure source term, and the forcing term representing the Coriolis effect. The vertical viscosity forcing term ($F_{\mu i}^{(\ell)}$) can be interpreted as a friction term between layers. To increase the solution stability, this term is treated implicitly. First, equation (4.1.1) is solved for each layer explicitly to obtain intermediate distribution functions $f_\alpha^{*(\ell)n+1}$, velocities $u_i^{*(\ell)n+1}$, and local water heights $h^{*(\ell)n+1}$:

$$f_\alpha^{*(\ell)n+1}\left(\mathbf{x}+\mathbf{c}_\alpha\Delta t\right)=f_\alpha^{(\ell)n}\left(\mathbf{x}\right)-\sum_{\beta=0}^{8}\frac{s_\beta \mathrm{b}_{\beta\alpha}}{\left\|\mathbf{b}_\beta\right\|}\left[m^{(\ell)n}{}_\beta\left(\mathbf{x},t\right)-m_\beta^{((\ell)eq)n}\left(\mathbf{x},t\right)\right]+\frac{\Delta t c_{\alpha i}}{6c^2}F_i^{(\ell)n}\left(\mathbf{x}\right) \tag{4.2.9}$$

$$h^{*(\ell)n+1}=\sum\nolimits_\alpha f_\alpha^{*(\ell)n+1} \tag{4.2.10}$$

$$u_i^{*(\ell)n+1}=\sum\nolimits_\alpha c_{\alpha i} f_\alpha^{*(\ell)n+1}\Big/ h^{*(\ell)n+1} \tag{4.2.11}$$

Then, an implicit step is solved for macroscopic variables given as:

$$\begin{bmatrix} h^{(\ell)n+1} \\ \left(hu_i\right)^{(\ell)n+1} \end{bmatrix}+\begin{bmatrix} 0 \\ F_{\mu i}^{(\ell)n+1} \end{bmatrix}=\begin{bmatrix} h^{*(\ell)n+1} \\ \left(hu_i\right)^{*(\ell)n+1} \end{bmatrix} \tag{4.2.12}$$

where the right hand side represents the intermediate solution given by the explicit LB step (equations (4.2.9)-(4.2.11)) and is known. The first row in equation (4.2.12) relates to mass conservation, which is not affected by the vertical viscosity forcing. The second row in equation (4.2.12) relates to momentum that needs to account for the vertical viscosity forcing. The updating of flow velocities $u_i^{(\ell)n+1}$ at time $n+1$ based on the second row in equation (4.2.12) leads to an $M\times M$ tridiagonal linear system (Audusse 2005):

$$\begin{bmatrix} a_1 & b_1 & 0 & \cdots & 0 \\ c_2 & a_2 & b_2 & \ddots & \vdots \\ 0 & c_3 & a_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{M-1} \\ 0 & \cdots & 0 & c_M & a_M \end{bmatrix} \begin{bmatrix} u_i^{(1)n+1} \\ u_i^{(2)n+1} \\ u_i^{(3)n+1} \\ \vdots \\ u_i^{(M)n+1} \end{bmatrix} = \begin{bmatrix} \left(hu_i\right)^{*(1)n+1} \\ \left(hu_i\right)^{*(2)n+1} \\ \left(hu_i\right)^{*(3)n+1} \\ \vdots \\ \left(hu_i\right)^{*(M)n+1} \end{bmatrix} \tag{4.2.13}$$

where the matrix elements are

$$\begin{aligned} a_1 &= h^{(1)n+1} + \frac{2\mu\Delta t}{h^{(1)n+1} + h^{(2)n+1}} + \kappa\Delta t \\ a_\ell &= h^{(\ell)n+1} + 2\mu\Delta t\left(\frac{1}{h^{(\ell)n+1} + h^{(\ell+1)n+1}} + \frac{1}{h^{(\ell)n+1} + h^{(\ell-1)n+1}} \right) \quad , \ell = 2,\ldots,M-1 \\ a_M &= h^{(M)n+1} + \frac{2\mu\Delta t}{h^{(M)n+1} + h^{(M-1)n+1}} \end{aligned} \tag{4.2.14}$$

$$b_\ell = h^{(\ell)n+1} + \frac{2\mu\Delta t}{h^{(\ell)n+1} + h^{(\ell+1)n+1}} \quad , \ell = 1,\ldots,M-1 \tag{4.2.15}$$

$$c_\ell = h^{(\ell)n+1} + \frac{2\mu\Delta t}{h^{(\ell)n+1} + h^{(\ell-1)n+1}} \quad , \ell = 2,\ldots,M \tag{4.2.16}$$

The above numerical procedure to advance the solutions of the multi-layer LB algorithm from time n to time $n+1$ can be summarized in the following steps:

Step 1. Use the local water heights, $h^{(\ell)n}$, total water depth, H^n and velocities, $u_i^{(\ell)n}$ at time n to compute the EDFs, $f_\alpha^{(\ell)eq}$, from equation (4.1.4).

Step 2. Calculate the distribution functions, $f_\alpha^{(\ell)}$, using equation (4.2.9) and impose bounce-back boundary conditions for each layer to ensure conservation of mass and momentum at the impermeable walls.

Step 3. Calculate intermediate local water heights, $h^{*(\ell)n+1}$, total water depth, H^{*n+1} and velocities, $u_i^{*(\ell)n+1}$ using equations (4.2.10) and (4.2.11).

Step 4. Update velocities $u_i^{(\ell)n+1}$ at time $n+1$ by solving the linear system in equation (4.2.13).

Step 5. Repeat Step 1 to Step 4 for next time step.

5 HPC LBM FOR SHALLOW WATER EQUATIONS VIA OPENMP

5.1 Introduction

To demonstrate the inherent parallelism feature of the LB model for higher performance computing, this study introduces OpenMP in a shared memory environment to test the scalability of the multi-layer LB code. OpenMP parallelization is a straightforward extension of serial coding, based on the simple recipe: i) add work distribution directives; ii) carefully control variable scoping; iii) if needed, choose the most suitable parallel loop scheduling; and iv) if needed, take care of false sharing of variables among threads. Although this approach works with some very simple codes, most programs parallelized in this way do not scale beyond a small number of processors (Massaioli and Amati 2002). A computational advantage of the LBM is that implementing streaming and colliding steps leads to a computational algorithm that is very suitable for parallelization in both shared and distributed memory environments. Massaioli and Amati (Massaioli and Amati 2002) implemented a two-dimensional LB algorithm using standard OpenMP directives and showed linear speedup to eight processors. The standard OpenMP directive uses an implicit control of the loop variable. In this study, implicit and explicit loop control approaches are compared in OpenMP for our multi-layer LB algorithm. The latter allows for cache optimization by dividing the computational domain of each processor into sub-domains.

5.2 Basic Code and Basic Parallelization

As mentioned above, the LBM implementation comprises two steps: streaming and collision. The collision step is completely local as there are essentially no spatial dependencies among variables and lends itself very favorably to parallelization. The streaming step, however has spatial dependencies of magnitude 1 and is the most critical part of the solution procedure as

far as the parallelization is concerned. Moreover, it takes up to approximately 25% of the total running time which would yield a theoretical limit for the speedup (Wellein et al. 2006). The basic code to be parallelized using OpenMP is written in FORTRAN 90 and follows the traditional practice of explicitly separating the collision and streaming operations in different subroutines so that the whole simulation can be pseudocoded as:

```
do from first time step to last time step
     call collision
     call streaming
enddo
```

5.3 Basic Parallelization

The pseudocode of the collision subroutine is given as:

```
sub collision
for each grid point (i, j)
    compute ω_α ( f_α^eq (i, j) − f_α (i, j) ) for every direction α
    update f_α (i, j)
end for each
```

The outermost loop, corresponding to the rightmost array index (i.e. to the *y* space coordinate) can be elected for parallelization, to reduce the overhead caused by entering/exiting work sharing constructs. A PARALLEL DO construct, with proper privatization of variables holding intermediate results, is enough. Each iteration has the same computational cost so that static scheduling can be used. No false sharing effect is present for non trivial grids. The non-locality of the streaming step requires more attention. To reduce memory occupancy, the streaming step is performed in place. As the pseudocode

```
sub streaming
for each direction α
     for each grid point (i, j)
          move f_α (i, j) to f_α (i + c_αi, j + c_αj)
     end for each
```

```
end for each
```

shows, different distribution functions move independently, but care must be taken depending on the length and direction of movement, so that the destination array element can be safely overwritten. Thus, every non zero component of c_α introduces a data dependency in the loop on the corresponding spatial index (loops are nested from greater to smaller stride, for cache efficiency reasons):

```
do j = nyf,nyi,-1
do i = nxi,nxf
```
$f_2(i,j) = f_2(i,j-1)$
$f_6(i,j) = f_6(i+1,j-1)$
```
enddo
enddo
```

```
do j = nyi,nyf
do i = nxf,nxi,-1
```
$f_4(i,j) = f_4(i,j+1)$
$f_8(i,j) = f_8(i-1,j+1)$
```
enddo
enddo
```

```
do j = nyf,nyi,-1
do i = nxf,nxi,-1
```
$f_1(i,j) = f_1(i-1,j)$
$f_5(i,j) = f_5(i-1,j-1)$
```
enddo
enddo
```

```
do j = nyi,nyf
do i = nxi,nxf
```
$f_3(i,j) = f_3(i+1,j)$
$f_7(i,j) = f_7(i+1,j+1)$
```
enddo
endd
```

5.4 Code Optimization and Parallelization

The streaming step is the obvious target for improving parallelization because it's data dependencies and memory access patterns. The streaming step may be attacked by noting that it contains a number of dependencies that prevent an efficient parallelization in its form; however, most of these difficulties may be overcome by alternating among two copies of distribution functions which alternate as source and destination on odd and even time steps. This approach is commonly accepted as good approach on both serial and parallel implementation. Furthermore, the approach allows one simple loop that can be iterated over for both streaming and collision. The resulting pseudo-code is given as:

```
do j=nyi,nyf
```

```
do i=nxi,nxf
```
$$f_1^{odd}(i,j) = f_1^{even}(i-1,j)$$
$$f_2^{odd}(i,j) = f_2^{even}(i,j-1)$$
$$f_3^{odd}(i,j) = f_3^{even}(i+1,j)$$
$$f_4^{odd}(i,j) = f_4^{even}(i,j+1)$$
$$f_5^{odd}(i,j) = f_5^{even}(i-1,j-1)$$
$$f_6^{odd}(i,j) = f_6^{even}(i+1,j-1)$$
$$f_7^{odd}(i,j) = f_7^{even}(i+1,j+1)$$
$$f_8^{odd}(i,j) = f_8^{even}(i-1,j+1)$$
```
enddo
enddo
```

A simple schedule achieves a good parallelization. Despite the increase in memory occupancy, there is a substantial benefit in the cache usage pattern providing an overall 15% serial speedup. The other technique used in this application has been to combine all phases, i.e. streaming, density and collision operators, into a single computational loop, so as to achieve maximal data reuse. With this modification, the code structure becomes:

```
do j=nyi,nyf
do i=nxi,nxf
```
$$f_1^{odd}(i,j) = f_1^{even}(i-1,j)$$
$$f_2^{odd}(i,j) = f_2^{even}(i,j-1)$$
$$f_3^{odd}(i,j) = f_3^{even}(i+1,j)$$
$$f_4^{odd}(i,j) = f_4^{even}(i,j+1)$$
$$f_5^{odd}(i,j) = f_5^{even}(i-1,j-1)$$
$$f_6^{odd}(i,j) = f_6^{even}(i+1,j-1)$$
$$f_7^{odd}(i,j) = f_7^{even}(i+1,j+1)$$
$$f_8^{odd}(i,j) = f_8^{even}(i-1,j+1)$$
compute $1/\tau\left(f_\alpha^{eq}(i,j) - f_\alpha^{odd}(i,j)\right)$ for every α

update $f_\alpha^{odd}(i,j)$
```
enddo
enddo
```

Massaioli and Amati (2002) found that a simple STATIC schedule for the PARALLEL DO directive was favorable and reported that runs with the basic LBM code structure with DYNAMIC and GUIDED schedules were not very satisfactory. Bella et al. (2002) suggested strategies to improve performance and efficiency based on implementing a fused approach using two copies of distribution function arrays and a static schedule alternating the source and destination arrays. This approach has been shown to improved parallelism by removing the spatial dependencies and has been proven to increase efficiency on both serial and parallel implementations. Bella et al. (2002) reported increased speedup and efficiency but only reported up to 6 processors. They also failed to report on how the size of the domain affects the parallel performance.

5.5 Cache Optimization

Obtaining data from the main memory in every computational cycle is time-consuming and the CPU remains idle during this process. If the data were being accessed from cache in every computational cycle, time consumed by data transfer would be much less. To obtain good performance on cache-based computer architectures, the computational algorithms are required to divide the data (computational domain) into blocks (sub-domains) that can fit into cache and then be utilized repeatedly. Not all algorithms are amenable to this kind of cache optimization since data dependencies disallow updating sub-domains separately. The lattice Boltzmann algorithm has only nearest neighbor data dependencies and is highly amenable to cache optimization.

5.6 Cache Optimization for LBM Using OpenMP

To implement the LBM algorithm efficiently with optimal use of cache, the grids need to be divided into subsections that fit in cache. Performing computations separately for each

subsection, for several time steps, achieves cache optimization. A standard approach of implementing the LBM algorithm is based on the OpenMP's PARALLEL region directive. The OpenMP PARALLEL DO directive uses an implicit loop control. Although simple to implement, the OpenMP PARALLEL DO directive has overhead caused by opening and closing of fork/join parallel regions each time step. It also has the disadvantage of not being able to control cache asscess. An explicit loop control technique is implemented to reduce the over head. This also has the added benefit of ensuring the same processor assignment in the parallel region and allows cache optimizations to be applied. This approach also is straightforward to extend to MPI programs for distributed memory systems. The domain is partitioned evenly over the i direction using an explicit loop schedule. This approach also keeps the same processor for each partition. Loop blocking is applied for cache optimization on each processor. This results in the final pseudo code shown below. The parallelization is load balanced for the periodic and simply boundary conditions considered in this study. OpenMP PARALLEL SECTIONS directive is used to ensure the processors stay synchronized inside a time step.

```
do jj=1,ny,jblksize
do ii=ibeg,iend,iblksize

do j=jj,min(ny, jj+jblksize-1)
do i=ii,min(iNx, ii+iblksize-1)
```

$$f_1^{odd}(i,j) = f_1^{even}(i-1,j)$$
$$f_2^{odd}(i,j) = f_2^{even}(i,j-1)$$
$$f_3^{odd}(i,j) = f_3^{even}(i+1,j)$$
$$f_4^{odd}(i,j) = f_4^{even}(i,j+1)$$
$$f_5^{odd}(i,j) = f_5^{even}(i-1,j-1)$$
$$f_6^{odd}(i,j) = f_6^{even}(i+1,j-1)$$
$$f_7^{odd}(i,j) = f_7^{even}(i+1,j+1)$$
$$f_8^{odd}(i,j) = f_8^{even}(i-1,j+1)$$

compute $1/\tau\left(f_\alpha^{eq}(i,j) - f_\alpha^{odd}(i,j)\right)$ for every α

update $f_\alpha^{odd}(i, j)$

enddo
enddo

enddo
enddo

There is more than one choice of domain decomposition for three dimensional data. The current parallel LB algorithm is implemented by decomposing the entire flow domain into several computational domains divided along one horizontal flow direction, i.e., the lateral flow direction, according to the number of processors. The choice of one horizontal flow direction is made to restrict data communication in one direction. This decomposition allows the LB equation to be calculated with minimal communication across processor domains to retain the parallel benefits of the LBM. The main key is that the domain is not decomposed in the vertical (layer) direction. This allows the LB equation and the implicit step to remain completely local in layers to each processor and does not require the tridiagonal solver for equation (4.2.13) to be parallelized. An added benefit is that the computational domain of each processor is further divided into sub-domains to allow for cache optimization.

5.7 Parallel Speedup and Efficiency

Parallel performance is often evaluated by two factors: parallel speedup and efficiency. The parallel speedup is the ratio of computation times for one thread to the total number of threads (processors) used. The efficiency is the average speedup over the total number of threads. These two factors are used to evaluate the HPC in the following numerical simulations of wind-driven circulation in a square lake of dimensions $64000\,m \times 64000\,m$ with a flat bottom. The multi-layer LB method was implemented on a shared memory HPC system, an AIX v5.3 constellation from IBM with 1.9 GHz IBM POWER5+ processors. The initial water depth is 10

m. The lake is discretized into a grid of size $1024\times1024\times10$ corresponding to 10 layers and $\Delta x = \Delta y = 62.5\, m$. The initial local water height is $1\, m$ for each layer and the initial flow velocity is zero. The LBM parameters are $\tau = 0.501$, $\Delta t = 8\,\mathrm{s}$, and $e = 7.8125\, m/s$. The physical parameters are $\tau_{iz}^{W} = 0.1\, N/m^2$, $\rho = 1025\, kg/m^3$, $\rho_a = 1.2\, kg/m^3$, $C_W = 0.0015$, $\mu = 0.01\, m^2/s$, $\kappa = 0.001\, m/s$, and $f_c = 0\, s^{-1}$. The wind direction is along positive x direction. Therefore, the wind velocity is $U_{Wx} = 7.4536\, m/s$ and $U_{Wy} = 0\, m/s$. The initial condition for the distribution functions are the EDFs in equation (4.1.4) with the static water ($\mathbf{u}^{(\ell)} = 0$) and the initial local water heights. Free-slip bounce-back boundary conditions are applied to the four vertical side walls of the lake. The simulations were run for 1000 time steps.

Table 5.1 shows the execution time, speedup and efficiency for the implicit and explicit loop controls that were tested up to 16 processors. It demonstrates the importance of cache optimization in LBM. Optimizing cache-use on each thread based on dividing the processor domains into sub-domains improves the speedup and efficiency. The explicit loop control implementation takes slightly longer than the implicit approach; however, the parallel performance of the implicit approach starts to break down past 12 processors. The speed up and efficiency of the implicit approach begins to decline after 8 threads while the explicit approach continues to scale up to 16 threads.

Table 5.1: Execution Time (min), Speedup, and Efficiency for Implicit and Explicit Loop Control Implementations on a 1024×1024×10 grid.

Threads	Implicit Loop Control			Explicit Loop Control		
	Execution Time (min)	Speedup	Efficiency	Execution Time (min)	Speedup	Efficiency
1	40.408	1.00	1.00	45.978	1.00	1.00
2	20.260	1.99	0.99	22.156	2.00	1.00
4	10.096	4.00	1.00	11.698	3.99	0.98
8	5.468	7.39	0.92	6.005	7.54	0.94
12	4.335	9.32	0.78	3.946	11.65	0.97
16	6.306	6.40	0.40	3.076	14.94	0.93

6 HPC LBM FOR SHALLOW WATER EQUATIONS VIA GPU ACCELERATION

6.1 Introduction

The computational advantage of the LBM is that implementing streaming and colliding steps leads to a computational algorithm that is very suitable for parallelization on both GPU and CPU based architectures. However, the implementation details and techniques on these two architectures for code optimization are completely different. Increasing CPU parallel performance relies on optimal data caching (Tubbs and Tsai 2009) when increasing parallel performance on GPUs relies on memory access patterns and latency (Tolke and Krafczyk 2008; Walsh et al. 2009).

The GPU is specialized for graphics rendering tasks, which require computationally-intensive, highly parallel computations with much more transistors devoted to data processing compared to traditional CPUs. More specifically, the GPU is especially well-suited to address problems that can be expressed as data-parallel computations with high arithmetic intensity (NVIDIA 2008). The data-parallel computations require the same program instructions to execute on many data elements in parallel. The arithmetic intensity is the ratio of arithmetic operations to memory operations.

Modern GPUs contain hundreds of arithmetic units (stream processors) to provide tremendous acceleration for numerically intensive scientific applications. The latest generation of high-end video cards offer considerable computing power using their 100-200 on-card processors, 0.3-1.0 plus GB of RAM and fast inter-processor communications. Recently, the use of the GPU to accelerate non-graphics computations has drawn much attention (Bolz et al. 2003; Kruger and Westermann 2003; Buck 2005). This is due to the fact that the computational power of GPU's has exceeded that of PC-based CPUs by more than one order of magnitude while being

available for a comparable price. Furthermore, GPU's have become a low cost high-performance computing (HPC) solution as GPU video cards have become standard in most desktop and laptop computers, and can eliminate the cost of expensive cluster computing alternatives for HPC tasks.

One promising application of this GPU computing capability is through MATLAB and MATLAB mex functions. With a properly developed mex function, the MATLAB user friendly interface can be used to perform behind the scenes parallel computations on the GPU. The GPU becomes a co-processor for the personal computer. While the MPI implements a means of parallel computing on a cluster of PC's, NVIDIAs's "Compute Unified Device Architecture" (CUDA) implements a means of computing on the large number of processors of a GPU. MPI is employed for complicated computations, but is often limited by inter-computer communications, while CUDA is employed for massive number of simple computations using the fast communication between many processors. Both approaches have their strengths and weaknesses. The LBM and MATLAB performance on vectorized codes appear to be ideally suited for GPU computation.

Computation on a GPU is basically a three step process: (1) Copy data to the GPU memory, (2) Execute code (the "kernel) to process that data, and (3) Copy the results back from the GPU memory. In general, code should be designed to minimize steps (1) and (3), which frequently limit the overall speed of the calculations. CUDA calculations usually start to outperform ordinary CPU calculations for large-size problems. GPU development is still in its infancy and the techniques and approaches may be still raw but promising. As interest continues to peak, more tools and functionality should become available.

To investigate the LBM code performance on GPU architectures for shallow water flow and mass transport, this study uses the Jacket GPU engine for MATLAB® on a single GPU

workstation. The Jacket GPU engine for MATLAB® is built on the Compute Unified Device Architecture (CUDA), which is a hardware and software architecture for issuing and managing computations on the GPU as a data-parallel computing device without the need of understanding and programming graphics rendering languages (NVIDIA 2008). CUDA technology enables the GPU to solve computationally intensive numerical modeling applications in a simplified programming interface, making it more accessible to computational scientists. Jacket is a GPU engine for MATLAB® built on NVIDIA's® CUDA technology. It enables standard MATLAB® code to run on the GPU. Jacket is a complete and transparent system, automatically making memory transfer and execution optimization decisions (Accelereyes 2009). Jacket GPU parallelization is a straightforward extension of serial CPU coding in MATLAB®, based on the simple recipe: i) vectorize MATLAB® code; ii) minimize memory transfers between host (CPU) and device (GPU); and iii) identify and use the appropriate processor for serial (host) vs. parallel (device) computations (Accelereyes 2009). In this study, the parallel performance of our LB algorithm and how it scales with increasing problem size is demonstrated using Jacket. This study specifically implements LBM on NVIDIA GPU architectures.

6.2 NVIDIA GPU Platform

The simulations conducted in this study are performed using a single workstation 3.0 GHz Intel® Core™2 Extreme quad core with an NVIDIA® Tesla™ C1060 Computing Processor. The NVIDIA® Tesla™ C1060 Computing Processor contains 240 stream processors running at 1.3 GHz, which has a peak performance of 933 GFLOPs. While the memory clock is 800 MHz and the memory size is 4 GB, the memory interface is 512 bit and the memory bandwidth is 102 GB/sec. The operating system used is Microsoft Windows XP. MATLAB®

version 7.7.0.471 (R2008b) is used along with Jacket version 1.2. This version of Jacket uses NVIDIA's CUDA version 2.3.

The internal structure of a Tesla™ C1060 Computing Processor consists of 30 multiprocessors (NVIDIA 2008), with eight stream processors per multiprocessor. Each multiprocessor has 16 KB of shared memory accessible by the eight stream processors. Computations on the GPU are organized into kernels (GPU programs) to be executed by multiple threads in parallel. Threads are organized into groups called thread blocks. All threads within a thread block execute the same kernel and communicate with each other through local multiprocessor shared memory. They synchronize their computation with built-in synchronization instructions. Usually, multiple thread blocks are employed because the hardware places constraints on the maximum number of threads in one single block. Thread blocks cannot synchronize executions as easily as threads within a single block can, nor do thread blocks communicate between their local shared memories. To communicate, the thread blocks must access global device memory on the GPU card. This constraint limits thread-to-thread communication and restrains the amount of work that can be done in one kernel invocation.

Although the GPU architecture places constraints on inter-thread communication, it is specifically designed to optimize the throughput of a single set of instructions operating simultaneously on a large number of data sources. To obtain good performance on GPU-based computer architectures, the computational algorithms based on structured grids or data are required to divide the data (computational domain) into blocks (sub-domains) that can fit into thread blocks and maximize memory access patterns to achieve high throughput. The LB algorithm has only nearest neighbor data dependencies and is highly amenable to GPU architectures.

6.3 NVIDIA CUDA

The NVIDIA CUDA technology is a fundamentally new computing architecture that enables the GPU to solve complex computational problems. CUDA (Compute Unified Device Architecture) technology gives computationally intensive applications access to the processing power of NVIDIA graphics processing units (GPU's) through a new programming interface. Software development is strongly simplified by using the standard C language. The CUDA Toolkit is a complete software development solution for programming CUDA-enabled GPU's. The Toolkit includes standard FFT and BLAS libraries, a C-compiler for the NVIDIA GPU and a runtime driver. CUDA technology is currently supported on the Linux and Microsoft Windows XP operating systems.

6.4 AccelerEye's Jacket

Jacket is a GPU engine for MATLAB that enables standard MATLAB code to run on the GPU, connecting the user-friendliness of MATLAB directly to the speed and visual computing capability of the GPU (Accelereyes 2008). MATLAB GPU computing with Jacket starts at the most basic level through the replacement of low-level MATLAB data structures which normally reside on the CPU with data structures that reside on the GPU. This replaces the lowest level of MATLAB's CPU-bound computation engine, moving the work MATLAB would normally do on the CPU to the GPU. Jacket is built on NVIDIA's CUDA technology. Jacket Beta version 0.3-20080710 on a 32 bit Windows XP with MATLAB R2007b is used. Jacket is run on the 2.0 beta version of the CUDA toolkit for Windows XP, which uses version 1.1 compute capabilities.

Jacket-enabled MATLAB scripts achieve speed improvements in the range of 2x - 10x improvements, and in some cases up to 100x improvements, over equivalent CPU versions. While Jacket accelerates MATLAB functions and computations at a lower level, the overall

speedup of an algorithm depends on the nature of the algorithm. The LBM has a very simple implementation consisting of only local calculations (collisions) and nearest neighbor memory transfers (streaming), which makes it a great candidate to be implemented both on the GPU and in MATLAB.

6.5 Optimizing MATLAB GPU Performance

Implementing algorithms on the GPU using Jacket requires certain considerations to optimized performance. Both MATLAB and Jacket perform best on vectorized code. A vectorized code can make it easy to determine which parts of an algorithm is inherently serial and parallel. Both MATLAB and Jacket take advantage of the inherent parallelism of the MATLAB's scripting M-language which is extremely powerful when utilized wisely. A good understanding of the memory dependencies of an algorithm is necessary as CPU's are inherently serial computing devices and GPU's are inherently parallel computing devices. For small or serial operations, computations on the CPU are likely to outperform computations on the GPU. For large or parallel operations, the GPU is likely to outperform the GPU. In a program, one can control which segments of code are run on each device through the casting operations. Each casting operation to and from the GPU pushes or pulls data back and forth from CPU memory to GPU memory. Excessive memory transfers should be avoided as it will reduce the performance of an application. The Jacket software minimizes these memory transfers automatically in normal operation. However, care must be taken in implementing an algorithm. Fortunately, the LBM can be completely vectorized and therefore all computations can be carried out on the GPU. Transfers to CPU memory are only necessary for outputting solutions at desired intervals. Currently a transfer to CPU is necessary for MATLAB plotting routines, however, due to the nature of the GPU plots can be created through OpenGL.

6.6 Computational Aspects

The basic code to be parallelized on the GPU using Jacket is written in MATLAB's M-Language and follows the same traditional practice of explicitly separating the collision and streaming operations. The solution algorithm has not changed. However, in order to take advantage of the GPU and MATLAB, the codes must be vectorized. Due to vectorization, the solution procedure focuses on three main steps: the calculation of local macroscopic variables from distribution functions, the collision step and the streaming step. Two copies of the distribution functions are necessary to allow the code to be vectorized. Other than that, the computations and procedures remain the same as the FORTRAN 90 code. Again, the whole simulation can be pseudo-coded as:

```
For time =first time step: last time step
     Compute Macroscopic Variables
     Collision Step
     Streaming Step
End
```

The vectorized version of the code is straightforward and very simple. The Jacket GPU engine makes translating the code on the GPU as simple as casting the variables to the GPU. From there, all calculations are performed on the GPU. Since the LBM is inherently parallel, there is no need to cast variables back to the CPU until the end of the simulation or when variables are written to file.

6.7 Parallel Performance

The parallel performance on the GPU is investigated in this section. Parallel speedup is used to evaluate the HPC in the following numerical simulations of wind-driven circulation in a rectangular lake of dimensions $170\,km \times 60\,km$ with a flat bottom. The parallel performance of the GPU is based on arithmetic intensity and data access patterns (NVIDIA 2008); therefore, the

parallel performance is investigated based on how the speed up scales with increasing problems size. The multi-layer LB method was implemented on a single workstation in MATLAB. The initial water depth is 10 m. The lake is discretized into a grids of size $10\times171\times61$, $10\times341\times121$, $10\times681\times241$, and $10\times1361\times481$ corresponding to 10 layers and $\Delta x = \Delta y$=1000 m, 500 m, 250 m, and 125 m , respectively. The initial local water height is $1\,m$ for each layer and the initial flow velocity is zero. The LBM parameters are $\tau = 0.501$ and $c = 20\ m/s$, with Δt calculated as $\Delta t = c/\Delta x$. The physical parameters are $\tau_{iz}^{W} = 0.1\ N/m^2$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.01\ m^2/s$, $\kappa = 0.001\ m/s$, and $f_c = 0\ s^{-1}$. The wind direction is along positive x direction. Therefore, the wind velocity is $U_{Wx} = 7.4536\ m/s$ and $U_{Wy} = 0\ m/s$. The initial condition for the distribution functions are the EDFs in equation (4.1.4) with the static water ($\mathbf{u}^{(\ell)} = 0$) and the initial local water heights. Free-slip bounce-back boundary conditions are applied to the four vertical side walls of the lake. The simulations were run for a simulation time of 30 hours where steady state has been achieved. The average time per time step is investigated to make a fair comparison on computational effort.

Table 6.1 shows the grid size, execution time per time step and speedup for the GPU over a single core of the CPU in MATLAB. It demonstrates the importance arithmetic intensity in LBM performance on the GPU. If the number of lattice nodes is sufficiently high, the computations will out weigh the data access and communication yielding a high arithmetic intensity. The multi-layer LB algorithm yields approximately 2X speed up on the smallest number of lattice nodes with the maximum speedup of approximately 22X on the largest number of lattice nodes.

Table 6.1: The grid size, execution time per time step and speedup for the GPU over a single core of the CPU in MATLAB

Grid Size	CPU	GPU	Speedup
	Execution Time per time step(s)	Execution Time per time step (s)	
1	0.44	0.19	2.23
2	3.04	0.30	10.13
4	14.19	0.95	14.92
8	56.60	2.57	22.00

7 NUMERICAL EXAMPLES

7.1 Dam Break Flow Over A Forward Facing Step

A two-dimensional dam-break problem over a forward facing step is considered (Benkhaldoun et al. 2007). The computational domain is a rectangular channel that is 12 m long and 6 m wide. The dam site and the step are located at distance $\ell = 6$ m from the upstream boundary. The bottom bed is assumed to be frictionless. The bed elevation is 0 m for $x \le \ell$ and 1 m for $x > \ell$. The initial water depth is 5 m for $x \le \ell$ and 1 m for $x > \ell$. The initial velocity is zero everywhere.

This test example is interesting since it includes most of flow structures such as shocks, rarefaction waves and contact discontinuities. The present study is limited to subcritical flow. This problem is formulated using the shallow water equations with no viscosity terms included (Benkhaldoun et al. 2007). Therefore, the viscosity terms, which can be viewed in the case as numerical viscosity, should approach zero in order not to smear the solution near the shock. This requires the relaxation time parameter, τ, to be close to the limit of 0.5 in the SRT-LBM and requires the relaxation rates s_7 and s_8 approaching 2.0 due to equation (3.3.11) in the MRT-LBM. This test aims to compare the performance and stability of the proposed GLBE and LBGK for solving the shallow water equations with very small kinematic viscosity.

To investigate the stability and accuracy for different grid size, the domain is discretized into grids of 201×101, 401×201, 801×401, 1601×801, 2001×1001, 2401×1201, and 2801×1401 lattices. The corresponding lattice size for the first three grids is $\Delta x = \Delta y = 0.06$ m, $\Delta x = \Delta y = 0.03$ m, and $\Delta x = \Delta y = 0.015$ m, respectively. The last four fine grids have the same grid size $\Delta x = \Delta y = 0.0075$ m. Therefore, the domain is increased to keep the constant grid spacing. A constant lattice speed $c = 16$ m/s is used. To achieve a kinematic viscosity of

$v = 1\times10^{-6}\ m^2/s$, the relaxation time parameter in the SRT-LBM is calculated using equation (3.3.13): $\tau = 0.500003125$, $\tau = 0.50000625$, $\tau = 0.5000125$, and $\tau = 0.500025$ for the four different Δx. The relaxation rates $s_\alpha = 1/\tau$ were used for the BGK and $s_4 = s_6 = s_7 = s_8 = 1/\tau$, and $s_1 = s_2 = s_7 - 0.6$ for the MRT.

The shallow water equations were run up to time t = 0.5 s. The total execution time and speedup for CPU and GPU implementations of the MRT-LBM are shown in Table 7.1. For the smallest grid (201×101), the GPU code takes longer time than the CPU code. This is due to communication overhead between grid blocks on the GPU that is larger than the actual computational cost for small grids. As the grid size increases, the computational cost becomes larger than the communication overhead which results in larger speed up. For the largest grid (2801×1401), the CPU code executed in 2164.1 s while the GPU code executed in 91.11 s, resulting in 23.75 times speed up.

The water free surface along the center of the channel at time t = 0.5 s is illustrated in Figure 7.1. The results are compared for the BGK and MRT collision operators for grid spacing $\Delta x = \Delta y = 0.06$ m and $\Delta x = \Delta y = 0.03$ m. The exact solution is calculated using the procedure from (Alcrudo and Benkhaldoun 2001). It is clear that BGK solutions are unstable due to τ very close to the linear stability limit of 0.5 (Servan-Camas and Tsai 2009). Using MRT gives the ability to tune the relaxation rates, s_1 and s_2, corresponding to the non-conserved energy and energy squared moments. The spurious oscillations are suppressed in the MRT. In Figure 7.2, the water free surface ($h + z_b$), the water head ($h + z_b + \frac{u^2}{2g}$), the Froude number (Fr), and the water discharge (hu) of the MRT results are compared well to the exact solution. The results shown here compare favorably with several studies in the literature (see for example, (Benkhaldoun et

al. 2007), (Tseng 2004)). It is noted that errors in the discharge near the discontinuity are expected since the correct capturing of the water discharge is more difficult than the water height in this class of test cases (Benkhaldoun et al. 2007). Nevertheless, the MRT-LBM for shallow water is inherently a well-balanced scheme with correct forcing terms and does not exhibit large errors near the discontinuity.

Table 7.1: Total Execution Time (s) and Speedup for CPU and GPU Implementations of the MRT-LBM, Example 7.1.

Grid Size	Execution Time (s)		Speedup
	CPU	GPU	
201 × 101	1.71	3.72	0.46
401 × 201	10.15	7.88	1.29
801 × 401	80.54	16.55	4.87
1601 × 801	640.69	45.24	14.16
2001 × 1001	1005.50	54.18	18.56
2401 × 1201	1430.20	70.29	20.35
2801 × 1401	2164.10	91.11	23.75

7.2 Flow of Partial Dam Break

Dam break problems present an important flow phenomenon in civil engineering. The dam break problem is an important benchmark to validating shallow water solvers ability to correctly model shocks or hydraulic jumps that show occur in shallow water flows (Alcrudo and Garcia-Navarro 1993; Chaudry 1993; Ambrosi 1995; Fagherazzi et al. 2004). A partial dam break problem first presented in (Fennema and Chaudhry 1990) is considered. The partial dam-break presents rapid opening of a sluice gate with non-symmetric breach. The computational domain is

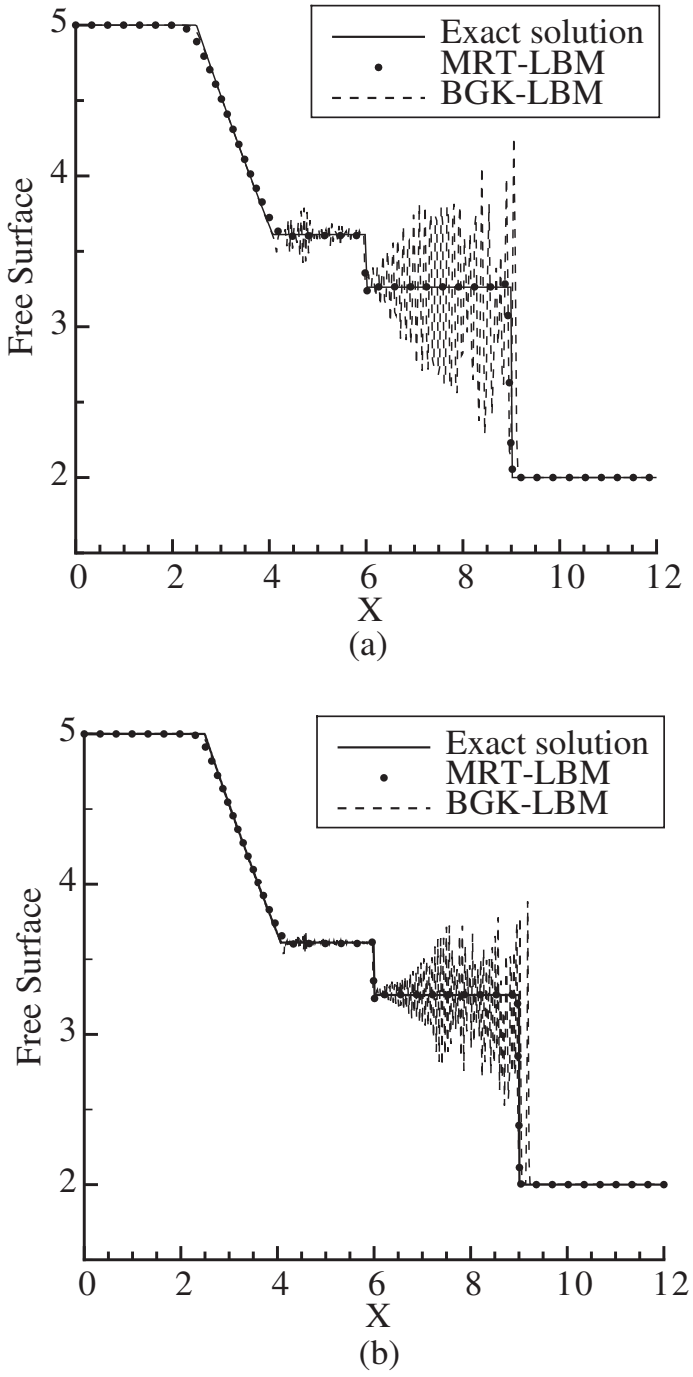


Figure 7.1: Free surface comparisons of subcritical flow over a forward facing step at time t = 0.5 s using the BGK and MRT collision operators for grid spacing (a) $\Delta x = \Delta y = 0.06$ m, and (b) $\Delta x = \Delta y = 0.03$ m.

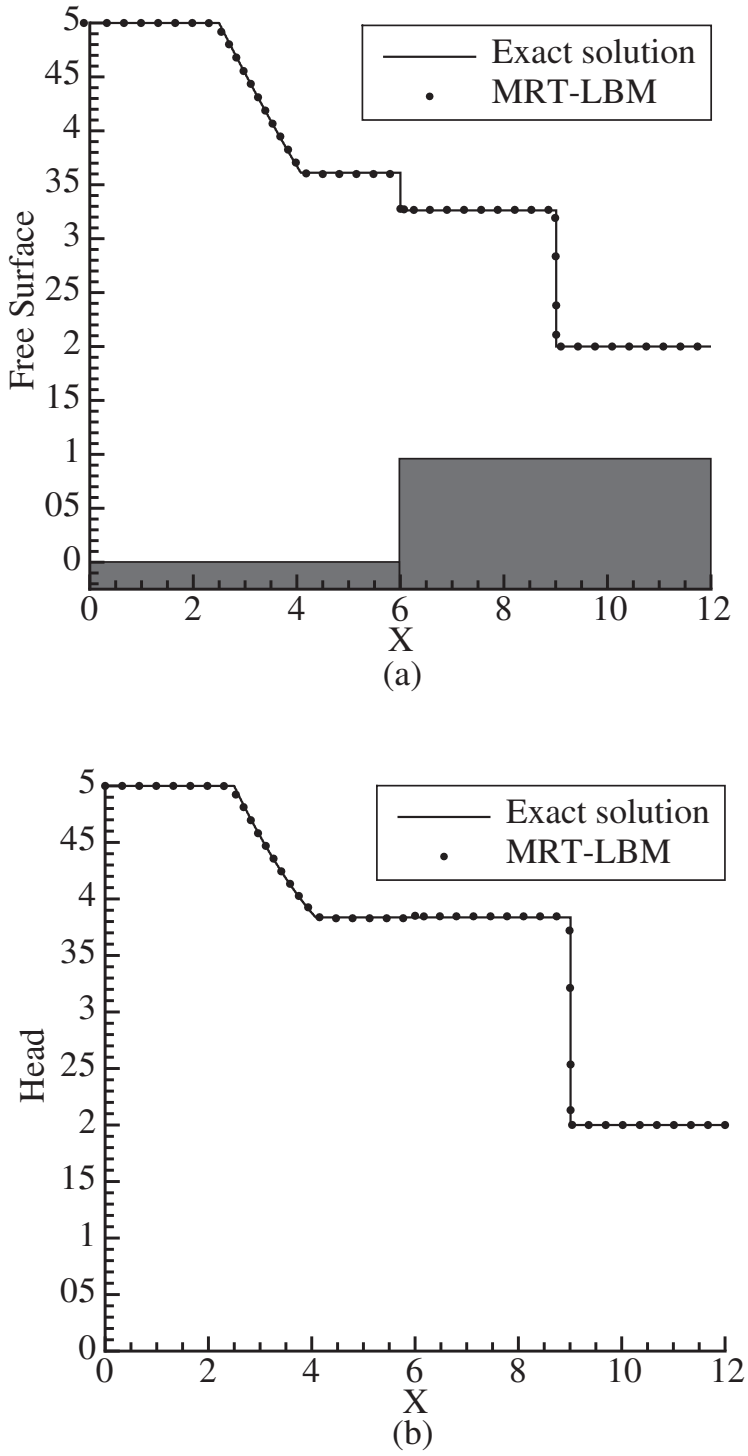
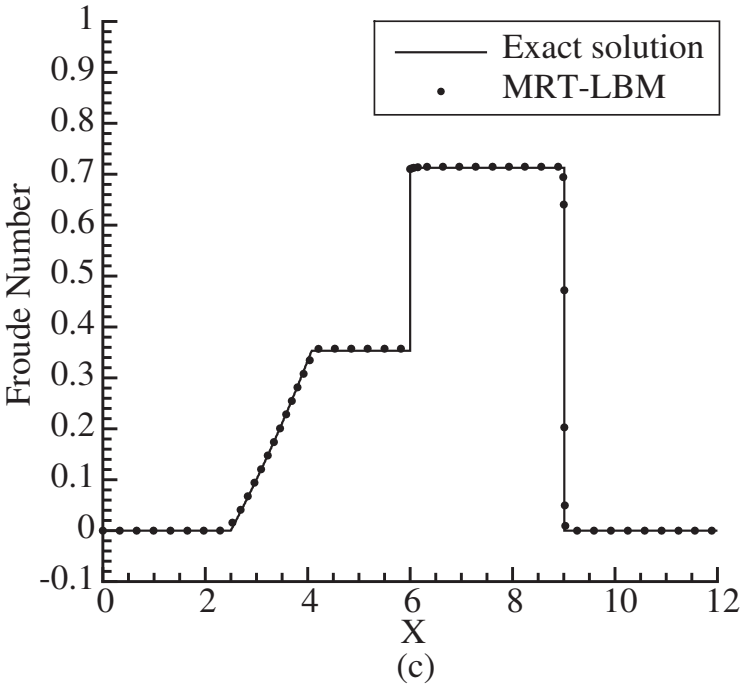


Figure 7.2: Comparisons of MRT-LBM simulation results to the exact solutions for subcritical flow over a forward facing step at time t = 0.5 s for (a) free surface, (b) water head.



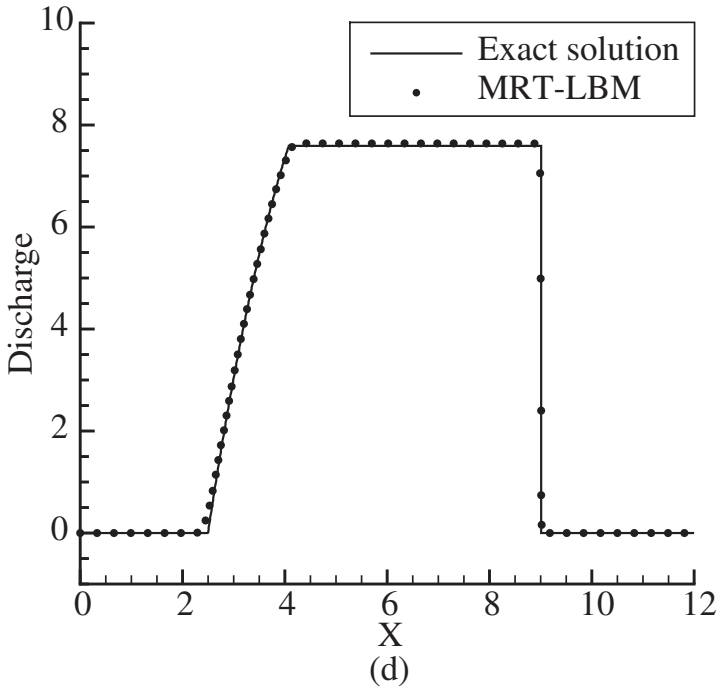


Figure 7.2: Comparisons of MRT-LBM simulation results to the exact solutions for subcritical flow over a forward facing step at time t = 0.5 s for (c) Froude number, and (d) discharge.

$200\ \text{m} \times 200\ \text{m}$, with a horizontal bottom. A dam, with 10m in thickness, is located in the middle of the domain. The initial upstream and downstream water depth is $7.5\ \text{m}$ and $5\ \text{m}$, respectively. The breach is $75\ \text{m}$ in length, which has distances of $30\ \text{m}$ from the left bank and $95\ \text{m}$ from the right.

The domain is discretized into a grid of size 201×201 lattices corresponding to $\Delta x = \Delta y = 1\ \text{m}$. A constant lattice speed $c = 50\ \text{m/s}$ is used. The single relaxation time parameter is $\tau = 0.5 + 3\times10^{-8}$ and the corresponding relaxation rate is $s_7 = 2.0 - 1.2\times10^{-7}$. To ensure stability and accuracy, the remaining relaxation rates are $s_1 = s_2 = s_4 = s_6 = 1.8$ and $s_8 = s_7$. Free slip (tangential) bounce back boundary conditions are applied to the walls. At $t = 0\,\text{s}$, the flow is at rest and the dam fails. Water is released through the non-symmetric breach.

The MRT-LBM results at $t = 7.2\,\text{s}$ are shown in Figure 7.3. A bore wave is formed that propagates downstream while spreading laterally. A depression wave moves upstream. Both waves are well resolved. The water depth and u_x profiles at $Y = 130\ \text{m}$ are shown in Figures 7.3a and 7.3b, respectively. The flow separates from the truncated dam walls downstream of the breach creating rotating eddies as shown in Figure 7.3c. The initial discontinuous condition presents challenges to numerical methods and is widely used to test its capability for discontinuous flows. The use of an asymmetric dam break problem will be able to validate that the correctly interacts with boundaries. This is important for shallow water flow problems with discontinuities when complex and irregular boundary conditions are encountered. Often, a high-resolution Riemann-solver based method is required to produce an accurate solution to the problem. Fagherazzi et al. (2004) used a discontinuous Galerkin (DG) FEM to model the partial dam break problem. There is no analytical reference solution for this test case, but in the literature, numerical results of various studies are available ((Fennema and Chaudhry 1990),

(Duan and Liu 2007)). Computed water surface elevation and flow filed compare favorably with the computed results of these studies. The results demonstrate the LBM ability to solve for the flow velocity field and discontinuous water surface without the need for a Riemann solver.

To illustrate the GPU performance on this problem, the grid resolution was increased 10 times in each direction to a grid size of 2001×2001 and run on the CPU and GPU. The average computing time per time step for grid size 201×201 was 0.022 s on the CPU and 0.037 s on the GPU. The average computing time per time step for grid size 2001×2001 was 2.49 s and 0.12 s on the CPU and GPU, respectively, resulting in a speedup of 21.63.

7.3 Mass Transport of Point Continuous Injection

The TRT-LBM code is verified on the two-dimensional mass transport in an infinite pool under uniform flow and depth with velocity-dependent dispersion coefficients. The concentration is continuously injected at a single point with a constant rate throughout the entire depth of the water column. The analytical solution for the concentration is (Fetter 1998):

$$C(x,y,t)=\frac{M}{4\pi\sqrt{D_{xx}D_{yy}}}\int_{\theta=0}^{\theta=t}\exp\left(-\frac{(x-\mathrm{U}_0\theta)^2}{4D_{xx}\theta}-\frac{y^2}{4D_{yy}\theta}\right)\frac{d\theta}{\theta}, \tag{8.1.1}$$

where M is the mass injection rate. U_0 is the uniform flow velocity in x-direction. $D_{xx}=h|\mathbf{u}|\kappa_L$ and $D_{yy}=h|\mathbf{u}|\kappa_T$ are obtained given $C_z=\sqrt{g}$ and $D_m=0$.

In the numerical experiments, water depth is $h=1$ m, the mass injection rate is $M=15$ kg m^2/s, and the uniform velocity is $\mathrm{U}_0=0.45$ m/s. The simulation domain was 3000 m $\times$ 1000 m. Given $\kappa_L=2.22$, the experiments were run for anisotropic dispersion ratios $D_{xx}/D_{yy}=5$, 10, and 25. Six different grid sizes are considered: 451×151, 901×301, 1801×601,

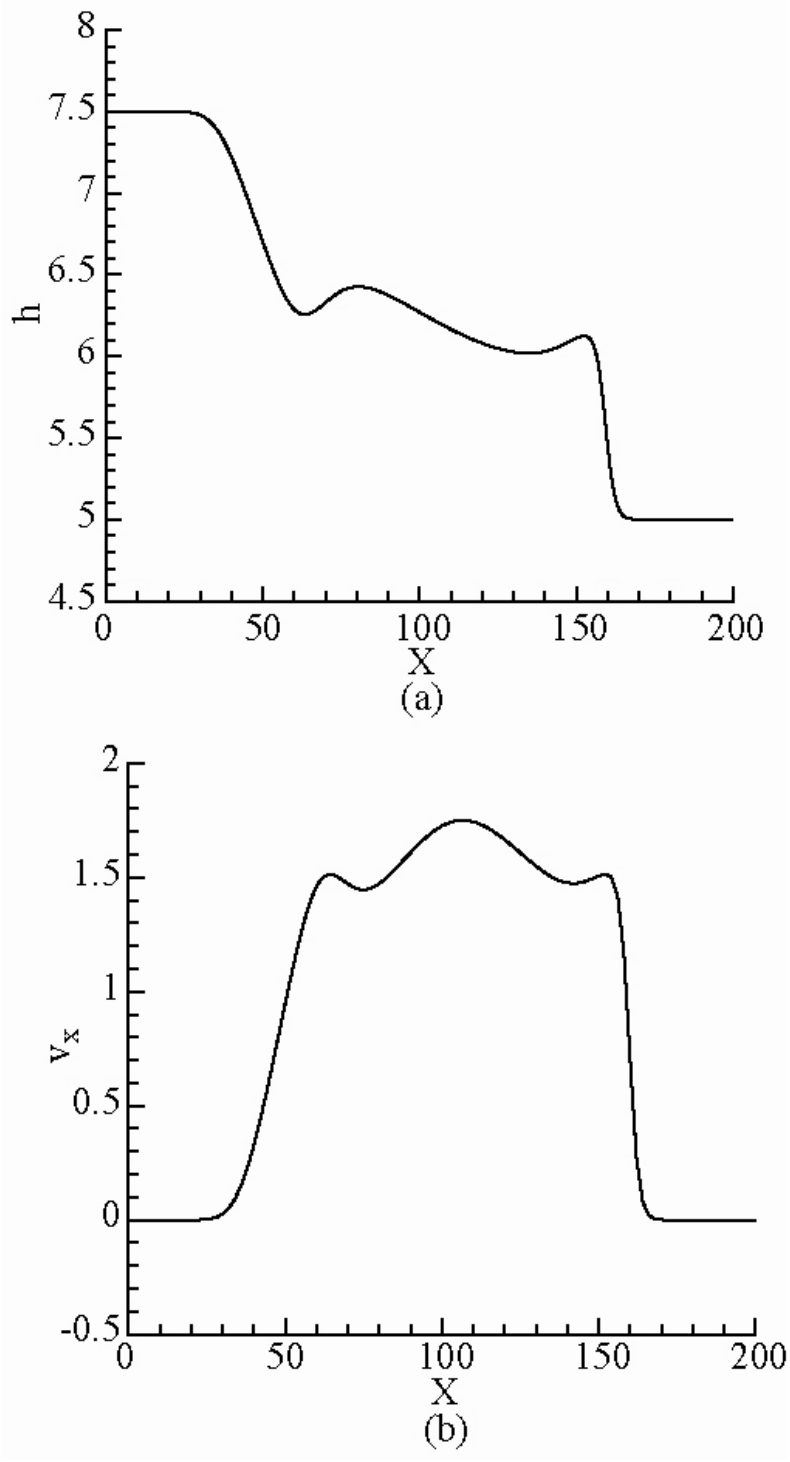


Figure 7.3: Two-dimensional partial dam break simulation results for $t = 7.2$ s. (a) water depth profile at $Y = 130$ m, (b) u_x profile at $Y = 130$ m.

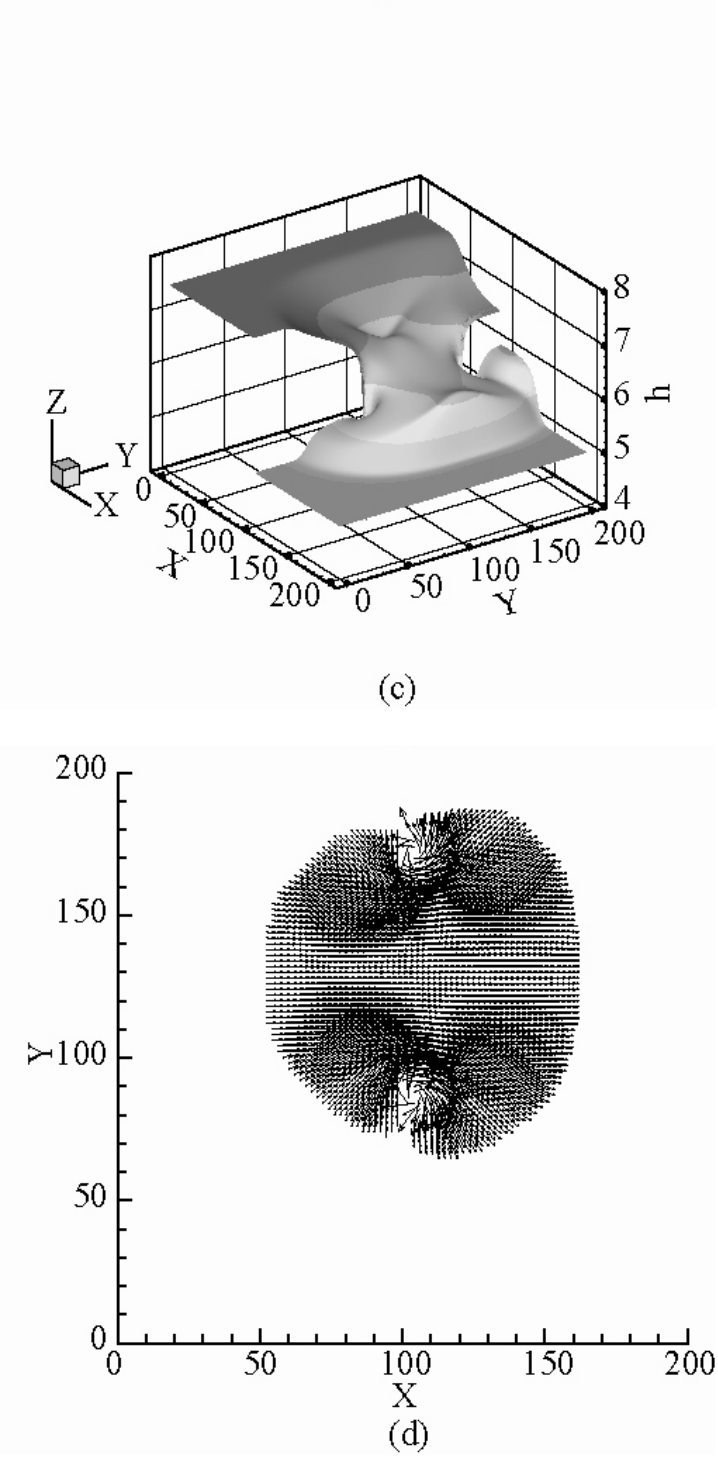


Figure 7.3: Two-dimensional partial dam break simulation results for $t = 7.2$ s. (c) water surface, and (d) velocity field.

2401×801, 3001×1001, and 3601×1201. The anti-symmetric relaxation time $\tau_a = 0.85$ is selected with the symmetric relaxation time parameter fixed to $\tau_s = 0.7381$.

Figure 7.4 shows the comparison of the TRT-LBM solutions along the x-axis and contours against the analytical solutions for grid size 3601×1201. The results show excellent agreement. Table 7.2 lists the average execution times per 1000 time steps for the TRT-LBM. The performance exhibits the same behavior with the speedup less than one for smaller grids and maximum speed up on the largest grid size. The average execution time per time step was 1.91 s and 0.11 s on the CPU and GPU, respectively, resulting in a speedup of 17.74. The speed up is smaller for the TRT-LBM for mass transport compared to the MRT-LBM for shallow water because the computational intensity is lower for the TRT-LBM.

Table 7.2: Average Execution Time (s) per 1000 Time Steps and Speedup for CPU and GPU Implementations of TRT-LBM, Example 7.2.

Grid Size	Execution Time (s)		Speedup
	CPU	GPU	
451 × 151	27.43	45.22	0.61
901 × 301	115.32	44.10	2.62
1801 × 601	481.12	48.45	9.93
2401 × 801	837.46	62.24	13.46
3001 × 1001	1300.43	83.50	15.57
3601 × 1201	1910.64	107.71	17.74

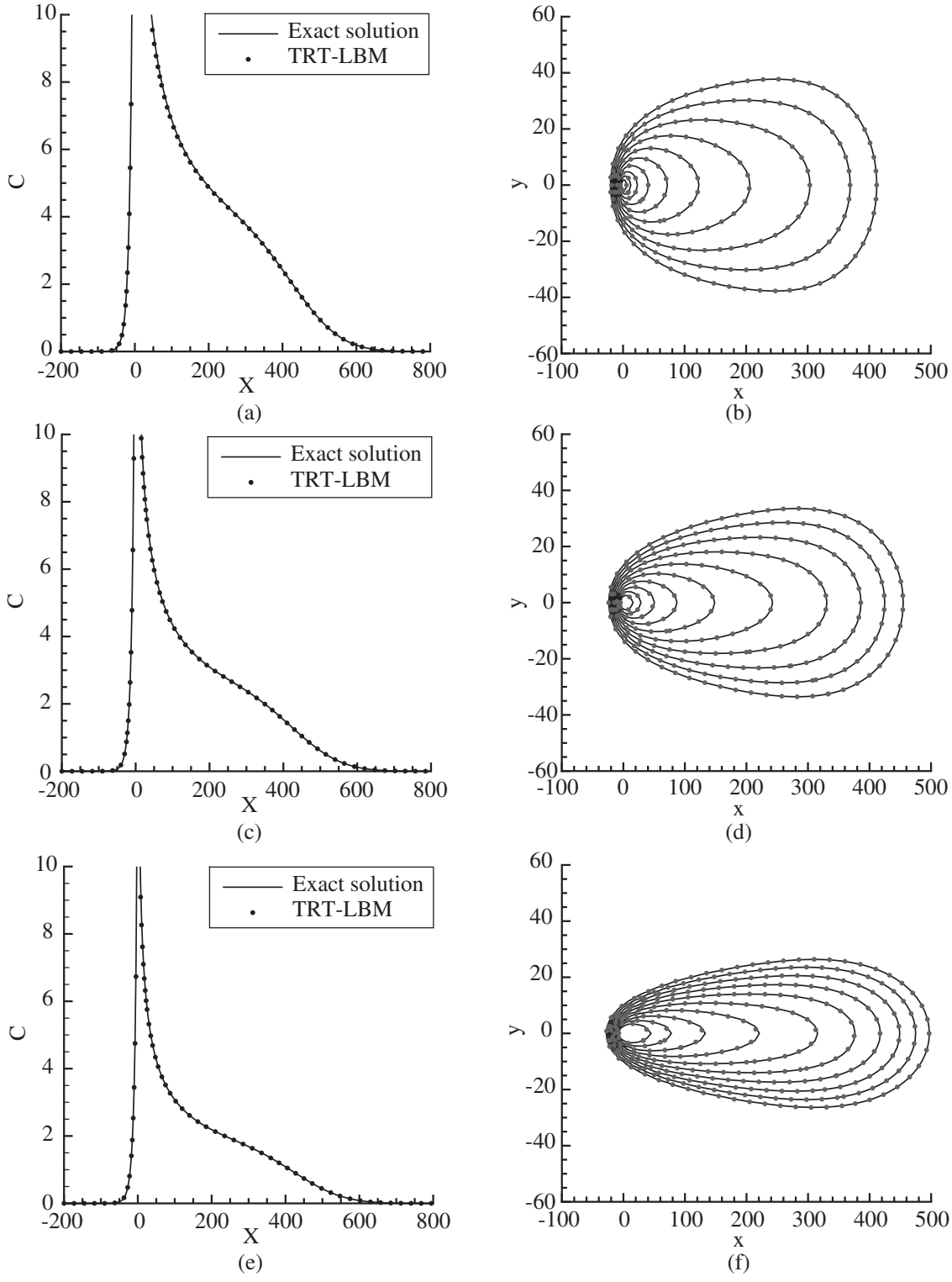


Figure 7.4: Concentration breakthrough curves and contours for anisotropy ratios: $\kappa_L / \kappa_T = 5$ for (a) and (b), $\kappa_L / \kappa_T = 10$ for (c) and (d), and $\kappa_L / \kappa_T = 25$ for (e) and (f).

7.4 Mass Transport in Partial Dam Break

The hypothetical partial dam break problem presented in section 7.2 is extended to include solute transport. The computational domain is $800\text{ m} \times 200\text{ m}$ with a horizontal bottom. The dam site is at $\text{x} = 100\text{ m}$. The initial upstream and downstream water depths are 10 m and 5 m, respectively. The initial upstream and downstream concentrations are 1 and 0, respectively. The breach is located in the same position as in section 7.2.

The domain is discretized into a grid of size 1601×401 lattices corresponding to grid spacing, $\Delta\text{x} = \Delta\text{y} = 0.5\text{ m}$. A lattice speed $c = 200\text{ m/s}$ is used. For the shallow water solver, a relaxation time parameter of $\tau = 0.5 + 3\times10^{-8}$ and the corresponding relaxation rate $s_7 = 2.0 - 1.2\times10^{-7}$ are used. To ensure stability and accuracy, the remaining relaxation rates are $s_1 = s_2 = s_4 = s_6 = 1.8$ and $s_8 = s_7$. For the transport solver, the anti-symmetric relaxation time parameter $\tau_a = 0.8$ and the symmetric relaxation time parameter $\tau_s = 0.7778$ are used. The longitudinal and transverse coefficients are $k_L = 5.93$ and $k_T = 0.23$ (Elder 1959). Free slip (tangential) bounce back boundary conditions for the flow problem and impermeable bounce back boundary conditions for the transport problem are applied to the walls. The initial flow is at rest.

The MRT-LBM results for water depths with grid spacing, $\Delta\text{x} = \Delta\text{y} = 0.5\text{ m}$, at $t = 10\text{ s}$, $t = 30\text{ s}$, $t = 60\text{ s}$, and $t = 90\text{ s}$ are shown in Figure 7.5. The maximum Peclet number is $Pe_{\max} = U_{\max}\Delta x / D_{yy} = 50$. Again, a bore wave is formed that propagates downstream while spreading laterally; and a depression wave moves upstream. The bore is allowed to propagate out of the domain with no reflection. The separated flow and rotating eddies propagate at a much slower rate than the bore. The TRT-LBM results for concentration with grid spacing,

$\Delta x = \Delta y = 0.5$ m, at $t = 10$ s, $t = 30$ s, $t = 60$ s, and $t = 90$ s are shown in Figure 7.6. A plume is formed that propagates downstream while spreading laterally. In the early stages, up to $t = 30$ s, the plume is transported due to the initial surge where the flow is yet dominated by downstream velocities. In the latter stages, after $t = 30$ s, the downstream velocities due to the surge begin to subside and the circulation velocities become dominated. At this point, the solute ceases to advect downstream and begins to mix in the lateral directions.

To investigate the GPU performance on the extended partial dam break problem, the domain was discretized into different grids: 401×101, 801×201, 1601×401, 2401×601, 3201×801, and 3601×901 lattices with aforementioned grid spacing, lattice speed and relaxation time parameters. Table 7.3 shows the execution time per 1000 times steps and speed up for combined MRT-LBE for shallow water and TRT-LBM for transport.

Table 7.3: Execution Time (s) per 1000 Times Steps and Speedup for CPU and GPU Implementations of MRT-LBM for Shallow Water and TRT-LBM for Transport, Example 7.3.

Grid Size	Execution Time (s)		Speedup
	CPU	GPU	
401 × 101	82	144	0.57
801 × 201	355	146	2.38
1601 × 401	1422	149	9.54
2401 × 601	3188	160	19.93
3201 × 801	5168	247	21.00
3601 × 901	6607	305	21.66

The performance exhibits the same behavior with speedup less than one for largest grid spacing and maximum speed up on the smallest grid spacing. For the smallest grid spacing, $\Delta x = \Delta y = 0.222$ m, corresponding to the largest grid size 3601×901, the average execution time per time step was 6.61 s on the CPU and 0.31 s on the GPU, resulting in a speedup of 21.66. The entire simulation took 32.66 hours on the CPU and 1.55 hours on the GPU for time up to 90 s.

7.5 Circulation in Rectangular Lake

The multi-layer LB model is verified using wind-driven, density-driven and a combination of wind- and density-driven circulation in a rectangular lake of dimensions $3400m \times 1400m$ with and a flat bottom. The initial water depth is 65 m, where an analytical solution for the horizontal velocity profile in depth is available (Shankar et al. 1997). The rectangular lake is discretized into 501×206 lattices in the planar direction corresponding to $\Delta x = 6.8\, m$ and $\Delta y = 6.8\, m$. The vertical direction is discretized into five, ten and twenty layers for each test case corresponding to an initial local water height of $8\, m$, $4\, m$, and $2\, m$ respectively. The LBM parameters are $\Delta t = 0.17$ s and $c = 40\, m/s$. To achieve a kinematic viscosity of $v = 1 \times 10^{-6}\ m^2/s$, the relaxation time parameter in the SRT-LBM is calculated using equation (3.3.13): $\tau - 0.5 = 3.8147 \times 10^{-7}$. The relaxation rates $s_4 = s_6 = s_7 = s_8 = 1/\tau$, and $s_1 = s_2 = s_7 - 0.6$ were used. The bed friction is based on a linear friction law. The initial conditions are applied by initializing the distribution functions to an EDF with the static water depth and local water heights, i.e. $f_\alpha^{(\ell)} = f_\alpha^{(\ell)eq}$, where $h^{(\ell)} = H^{init}/M$ and $\mathbf{u}^{(\ell)} = 0$. Free-slip bounce-back boundary conditions are applied to the four vertical side walls of the lake. The numerical simulation was carried out up to the establishment of a steady state two-dimensional circulation. The u_x velocity profile at the center of the lake,

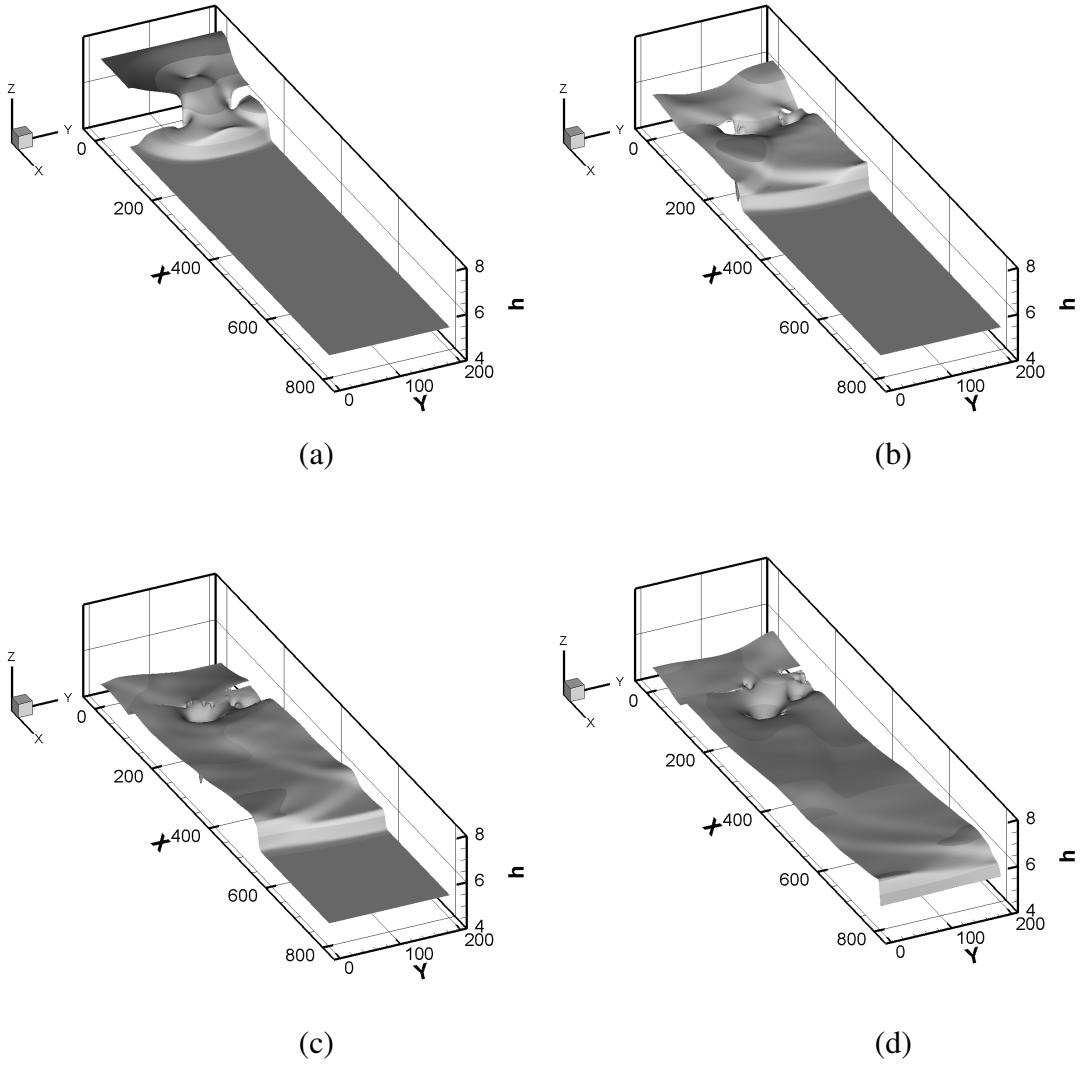


Figure 7.5: Water depth for extended partial dam break simulations at (a) t = 10 s, (b) t = 30 s, (c) t = 60 s, and (d) t = 90 s.

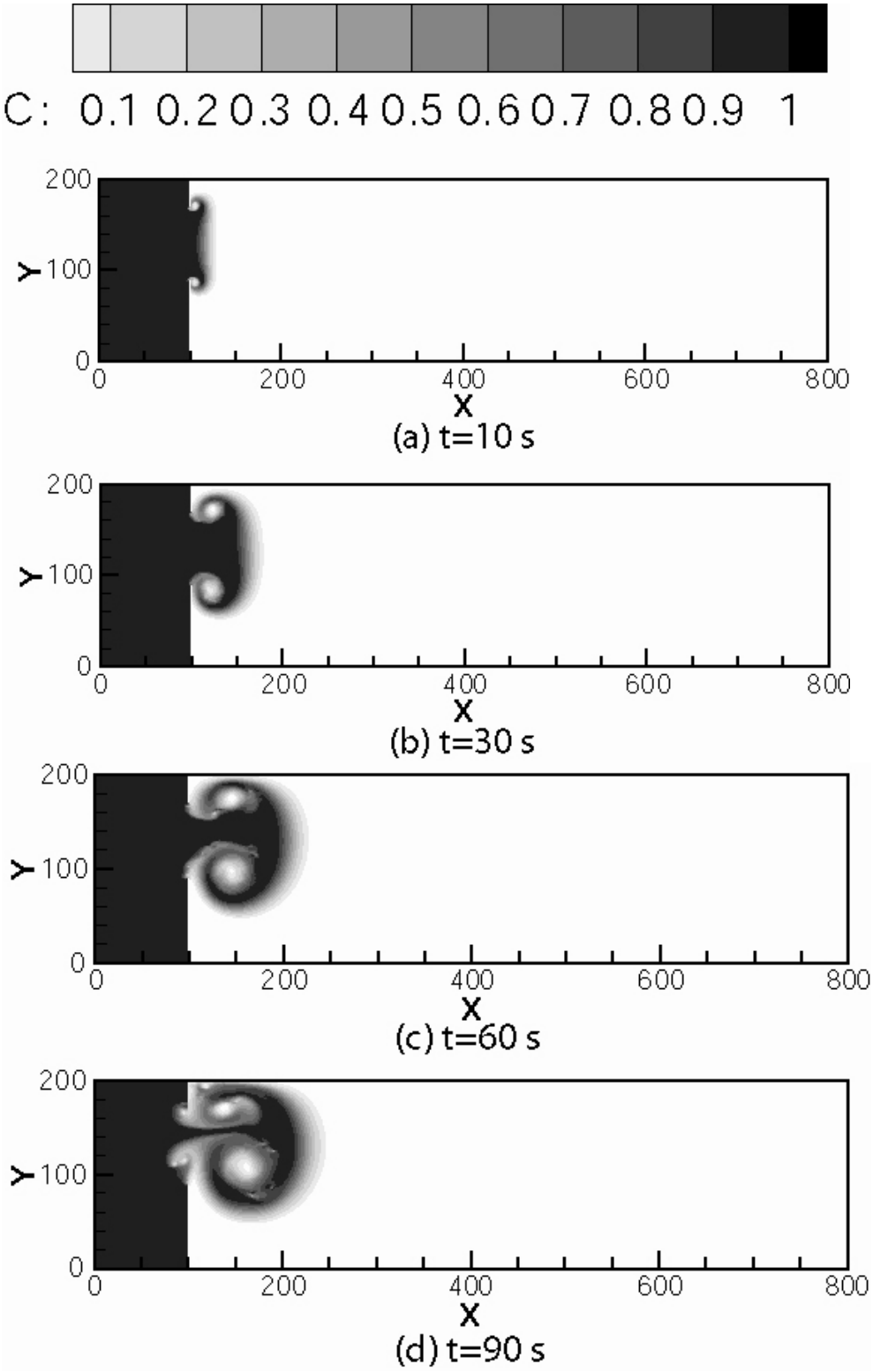


Figure 7.6: Concentration distributions for extended partial dam break simulations at (a) t = 10 s, (b) t = 30 s, (c) t = 60 s, and (d) t = 90 s.

$x = 1700$ m, $y = 700$ m, was compared against the analytical solution of Navier-Stokes equations assuming the surface slope and the horizontal velocity are constant in the longitudinal direction (Shankar et al. 1997). The vertical eddy viscosity is also assumed to be constant in the vertical direction.

The wind-driven circulation validation is performed for two different wind stress values, $\tau_{iz}^{W} = 0.03\ N/m^2$ and $\tau_{iz}^{W} = 0.3\ N/m^2$. The physical parameters for this case are, $\partial\rho/\partial x = 0$, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.005\ m/s$, The wind direction is along the positive x direction. The wind velocities are $U_{Wx} = 0.7071\ m/s$ and $U_{Wx} = 7.071\ m/s$, respectively, with $U_{Wy} = 0\ m/s$. The multi-layer LB solutions compare well to the analytical solutions of u_x profile for uniform wind stresses of $0.03\ N/m^2$ and $0.3\ N/m^2$, as shown in figures 7.7a and 7.7b.

The density-driven circulation validation is performed for two different horizontal density gradients, $\partial\rho/\partial x = -5\times10^{-7}\ kg/m$ and $\partial\rho/\partial x = -5\times10^{-5}\ kg/m$. The physical parameters for this case are $\tau_{iz}^{W} = 0$, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.005\ m/s$. The multi-layer LB solutions compare well to the analytical solutions of u_x profile for constant horizontal density gradients $\partial\rho/\partial x = -5\times10^{-7}\ kg/m$ and $\partial\rho/\partial x = -5\times10^{-5}\ kg/m$ as shown in figures 7.8a and 7.8b.

The multi-layer LB algorithm is also validated using a combination of wind- and density-driven circulation with $\tau_{iz}^{W} = 0.03\ N/m^2$, $\partial\rho/\partial x = -5\times10^{-5}\ kg/m$ and $\tau_{iz}^{W} = 0.3\ N/m^2$, $\partial\rho/\partial x = -5\times10^{-4}\ kg/m$. The physical parameters for this case are, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.005\ m/s$, The wind direction is along

the positive x direction. The multi-layer LB solutions compare well to the analytical solutions of u_x profile for the combined effects of wind- and density-driven circulation, as shown in figures 7.9a and 7.9b.

These examples demonstrate that the multi-layer LB model was capable of simulating wind-driven currents, density-driven current and the combination of wind- and density-driven flows.

7.6 Wind-driven Circulation in Rotating and Non-rotating Basins

In this example, the multi-layer LB model is demonstrated by simulating the wind-driven circulation with and without rotation over laterally varied bathymetric cross sections. A triangular bathymetry (see Figure 7.10) and two Gaussian bathymetry profiles are considered. The initial water depth for the triangular bathymetry has a minimum of $3\,m$ and maximum of $20\,m$. The Gaussian bathymetry profiles have initial water depths given by

$$H_0(y) = 8 + 12\exp\left[-\left(\frac{y}{2000}\right)^2\right] \tag{7.1.2}$$

and

$$H_0(y) = 8 + 8\exp\left[-\left(\frac{y-3D/10}{2000}\right)^2\right] + 12\exp\left[-\left(\frac{y+3D/10}{2000}\right)^2\right] \tag{7.1.3}$$

where D is the width of the basin. As shown in Figure 7.10, the x axis coincided with the southern lateral wall of the basin and pointed toward the head of the system. The y axis is laid along the closed boundary at $x = 0$. For each numerical example, the numerical domain consists of a longitudinally uniform basin 100 km long and 10 km wide for triangular and 15 km wide for Gaussian bathymetry profiles. Grid size is $250\,m$ along the x and y directions. Ten vertical layers are used.

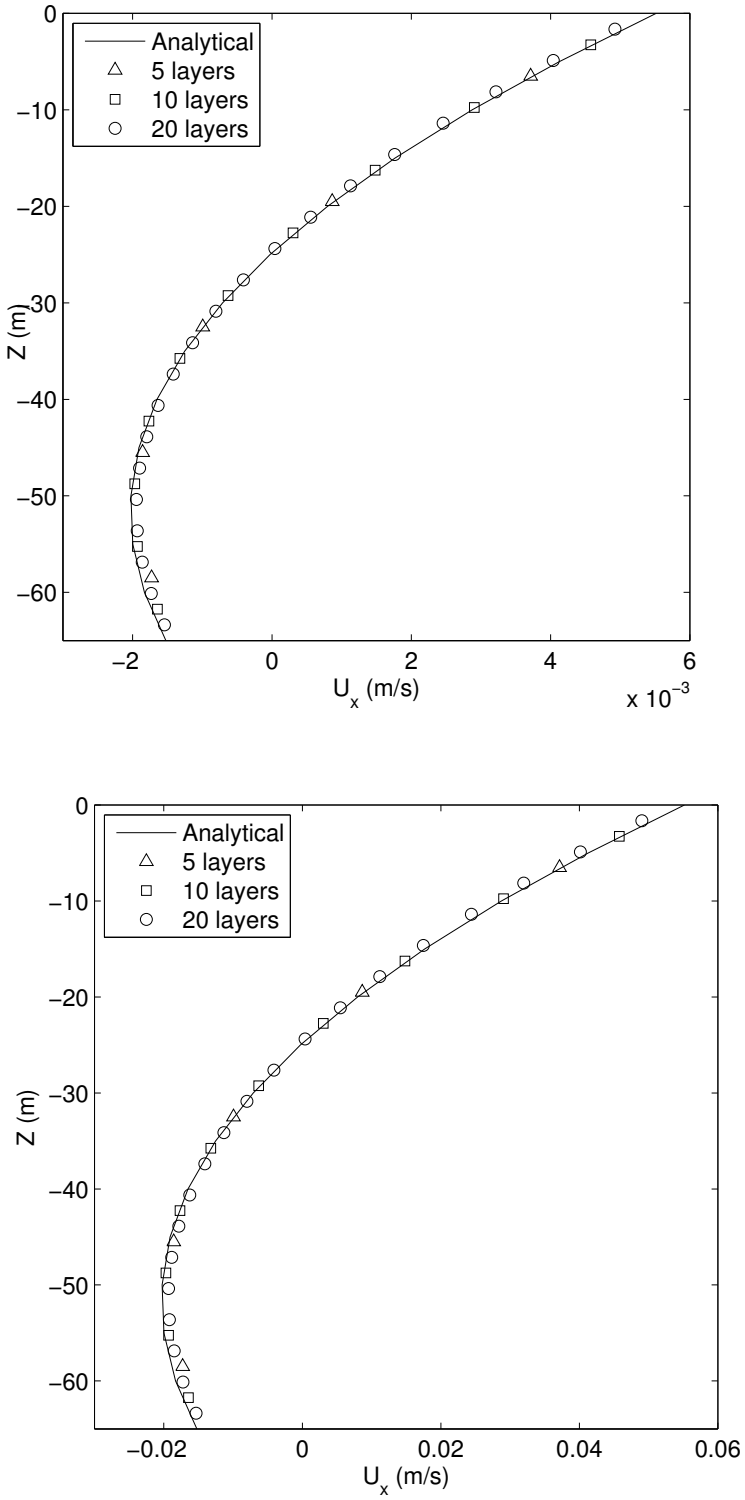
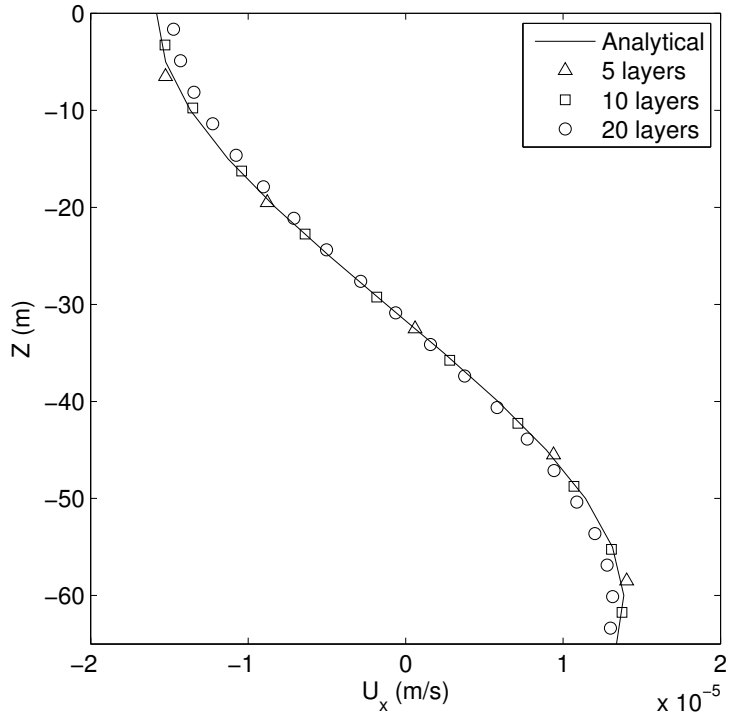


Figure 7.7: Comparisons of numerical model prediction with analytical solution for: (a) $\tau_{iz}^{W} = 0.03\ N/m^2$, and (b) $\tau_{iz}^{W} = 0.3\ N/m^2$.



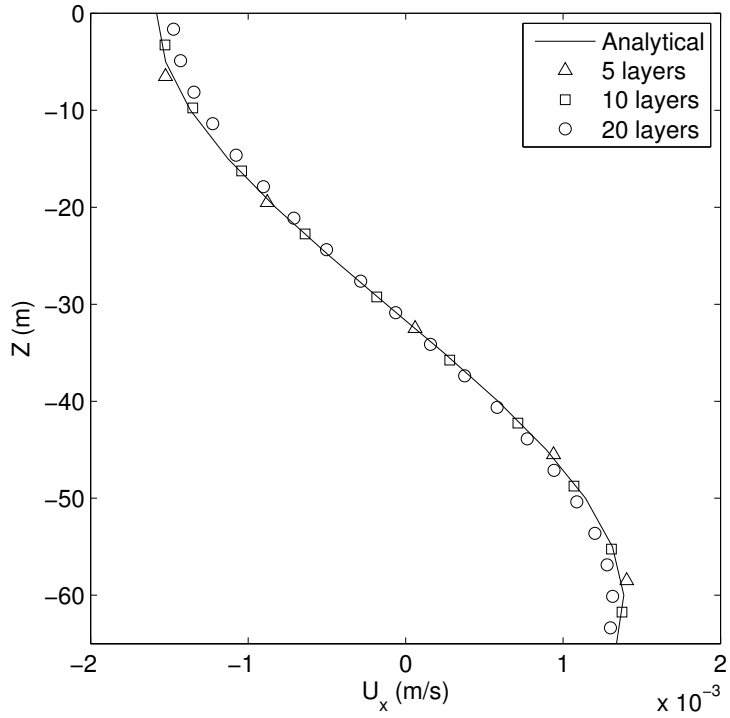


Figure 7.8: Comparisons of numerical model prediction with analytical solution for: (a) $\partial\rho/\partial x = -5\times10^{-7}\ kg/m$ and (b) $\partial\rho/\partial x = -5\times10^{-5}\ kg/m$.

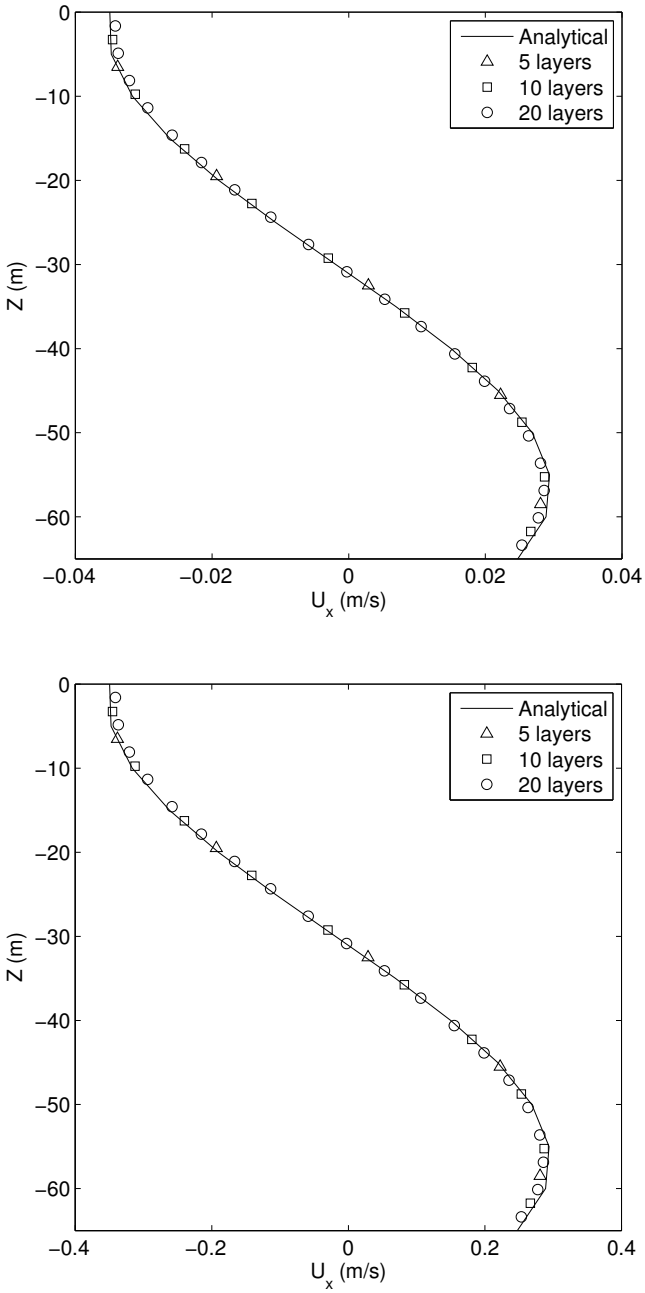


Figure 7.9: Comparisons of numerical model prediction with analytical solution for: (a) $\tau_{iz}^{W} = 0.03\ N/m^2$, $\partial\rho/\partial x = -5\times10^{-5}\ kg/m$, and (b) $\tau_{iz}^{W} = 0.3\ N/m^2$, $\partial\rho/\partial x = -5\times10^{-4}\ kg/m$.

In the LBM formulation, the computational domain is covered by $401\times41\times10$ lattices for the triangular bathymetry profile and $401\times61\times10$ lattices for the Gaussian bathymetry profiles with $\Delta x = 250\,m$, $\Delta t = 12.5\,s$, and $c = 20\,m/s$. To achieve a kinematic viscosity of $v = 1\times10^{-6}\,m^2/s$, the relaxation time parameter in the SRT-LBM is calculated using equation (3.3.13): $\tau - 0.5 = 6\times10^{-10}$. The relaxation rates $s_4 = s_6 = s_7 = s_8 = 1/\tau$, and $s_1 = s_2 = s_7 - 0.6$ were used. The free-slip condition was used for all closed boundaries. The initial water in the basin was static and a wind stress was increased linearly during the first six simulated hours. After six hours, the wind was constant. The wind stress acted along the positive x direction and blew uniformly throughout the domain. The numerical simulations were run up to 2 days after the wind stress was constant with the establishment of a steady state velocity field occurring after about 1 day. The numerical simulations were run on a single workstation with a 3.0 GHz Intel® Core™2 Extreme quad core processor.

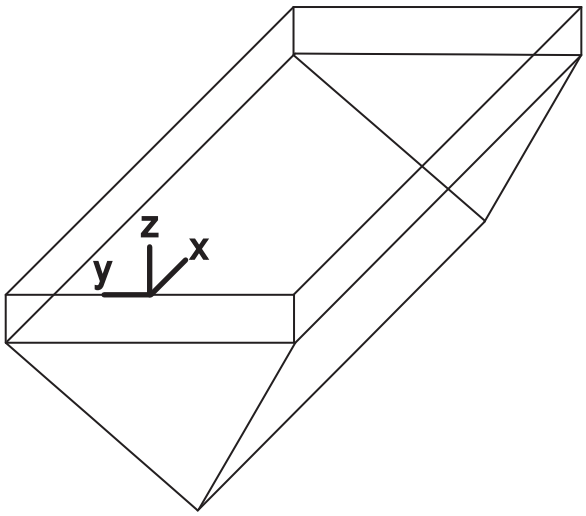


Figure 7.10: Computation domain with triangular bathymetry profile.

The first example is for the non-rotating case, in which the bathymetric profile was triangular with minimum depth of $3\,m$, maximum depth of $20\,m$. The physical parameters are $\tau_{iz}^{W} = 0.03\ N/m^2$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.0025\ m/s$, and $f_c = 0\ s^{-1}$. The wind stress was applied in the positive x direction. The wind velocity is $U_{Wx} = 4.0825\ m/s$ and $U_{Wy} = 0\ m/s$. The bed friction is formulated using a linear friction law. Figure 7.11 shows the u_x and u_y distributions at $x = 50\ km$ and $x = 98\ km$ planes. In the shallow region along the transverse boundaries in Figures 7.11a and 7.11b, u_x flows in the direction of the wind at all depths. The u_x flows in the opposite direction of the wind in the central part of the channel. The magnitude of the flow is highest near the surface and decreases with depth as expected from the bottom friction. Figures 7.11c and 7.11d show a divide of the transverse flow, u_y at $y = 0$. The symmetric distributions are due to neglecting Coriolis effect (non-rotating).

To further demonstrate the current method, the effect of the Earth's rotation with Coriolis parameter, $f_c = 10^{-4}\ s^{-1}$ is added to the first experiment. The u_x distributions in Figures 7.12a and 7.12b show similar lateral variability to those in the non-rotating case. The difference is seen at $x = 98\ km$ plane where u_x is symmetric for the non-rotating case, but asymmetric for the rotating case. The spatial distributions of u_x at $x = 50\ km$ and $x = 98\ km$ planes for both cases were consistent with those obtained by (Sanay and Valle-Levinson 2005). Due the Coriolis effect, the u_y distributions in Figures 7.12c and 7.12d are not symmetric.

The simulation time for both cases was 20.67 min on a single core and 5.07 min on four cores of a single workstation, demonstrating the expected 4 times speedup.

An important feature of the solution caused by the inclusion of the Earth's rotation is the free surface elevation distribution in the domain as shown in Figure 7.13. In the non-rotating case, the surface elevation distribution was only a function of along-channel direction. The across-channel barotropic pressure gradient expected in wind-driven flow over flat-bottom rotating systems is nearly a linear function of y location (Sanay and Valle-Levinson 2005). The combination of Coriolis effect and laterally varying bathymetry produced surface elevation contours with stronger lateral variability compared to the non-rotating case. This is due to the spatial variability in the longitudinal flow and vertical mixing dictated by the laterally varying bathymetry. The spatial distribution of the simulated surface elevation was consistent with that obtained by (Sanay and Valle-Levinson 2005) and (Glorioso and Davies 1995).

The second numerical example in the rotating case considers a Gaussian bathymetric profile with initial water depth given by equation (7.1.2). The initial water depth has a minimum of $8\,m$ and a maximum of $20\,m$ located at the center of basin. The physical parameter values remain the same, but the Coriolis parameter is $f_c = 10^{-4}\ s^{-1}$. The Gaussian profile produces a channel-shoal combination resulting in the flow patterns shown in Figure 7.14. The u_x flows in the direction of the wind for shallow regions along the transverse boundaries and in the opposite direction of the wind in the central part of the channel as shown in Figures 7.14a and 7.14b. The asymmetry near the closed boundary is more prevalent for this bathymetry profile (see Figure 7.14b). The u_y distributions shown in Figures 7.14c and 7.14d are asymmetric because of the Coriolis effect. The magnitude of u_y at $x = 98\,km$ plane is much larger than at $x = 50\,km$ plane.

The third numerical example of the rotating case considers a bathymetric profile with two Gaussians and initial water depth given by equation (7.1.3) with a minimum of $8\,m$. The

maximum depth is $20\,m$ located at $y=-3D/10\,km$, and local maximum depth is $16\,m$ located at $y=3D/10\,km$. Using the same physical parameter values as in the previous Gaussian bathymetric case, the flow patterns are shown in Figure 7.15. Similarly, the u_x flows in the direction of the wind in the shallow regions and flows in the opposite direction of the wind in both deeper channels as shown in Figure 7.15a and 7.15b. The u_y is very small at $x=50\,km$ plane (Figure 7.15c) while it is very strong at $x=98\,km$ plane (Figure 7.15d).

The simulation time for the second and third numerical examples was $32.89\,\text{min}$ on a single core and $8.12\,\text{min}$ on four cores of a single workstation, which also demonstrates the expected 4 times speedup.

7.7 Wind- and Density-driven Circulation in Rotating Basins

In this example, the multi-layer LB model is demonstrated on GPU-based HPC by simulating wind-driven, density-driven, and combined wind- and density-driven circulation over a Gaussian bathymetry profile. The Gaussian bathymetry profile has initial water depth given by equation 7.1.3. Again, the x axis coincides with the southern lateral wall of the basin and pointed toward the head of the system. The y axis is laid along the closed boundary at $x=0$. The numerical domain consists of a longitudinally uniform basin 100 km long and 10 15 km wide. The Grid spacing is $125\,m$ along the x and y directions. Ten vertical layers are used. The maximum depth is $27\,m$ located at $y=-3D/10\,km$, and local maximum depth is $11\,m$ located at $y=3D/10\,km$.

For each case, the LBM formulation consists of the computational domain covered by $801\times121\times10$ lattices with LBM parameters, $\Delta x=125\,m$, $\Delta t=6.25\,s$, and $c=20\,m/s$. To achieve a kinematic viscosity of $v=1\times10^{-6}\,m^2/s$, the relaxation time parameter in the SRT-

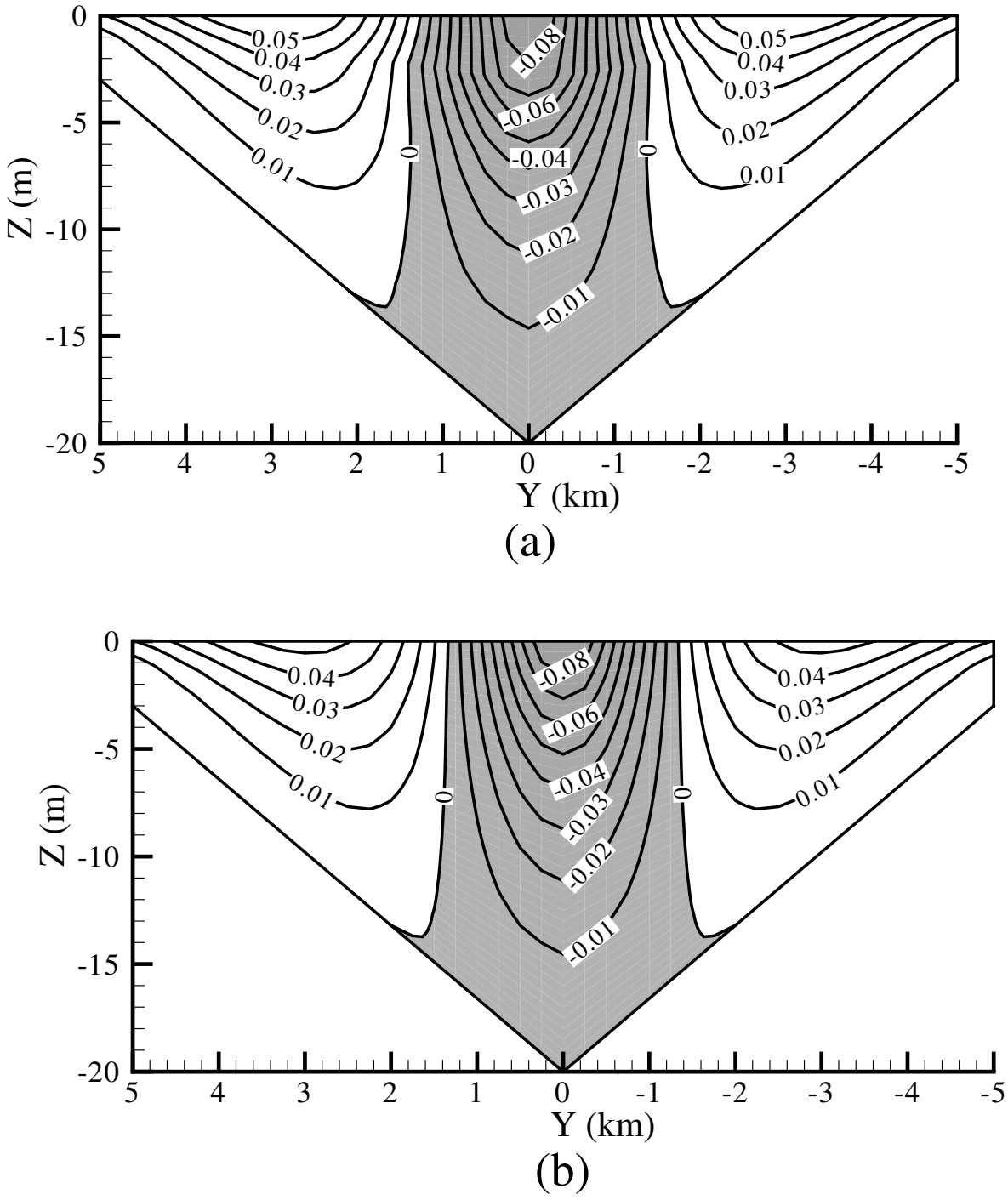


Figure 7.11: Contours of u_x velocity (m/s) at (a) $x = 50\ km$, and (b) $x = 98\ km$ for a non-rotating system with the triangular bathymetry profile. The dark areas represent negative velocities

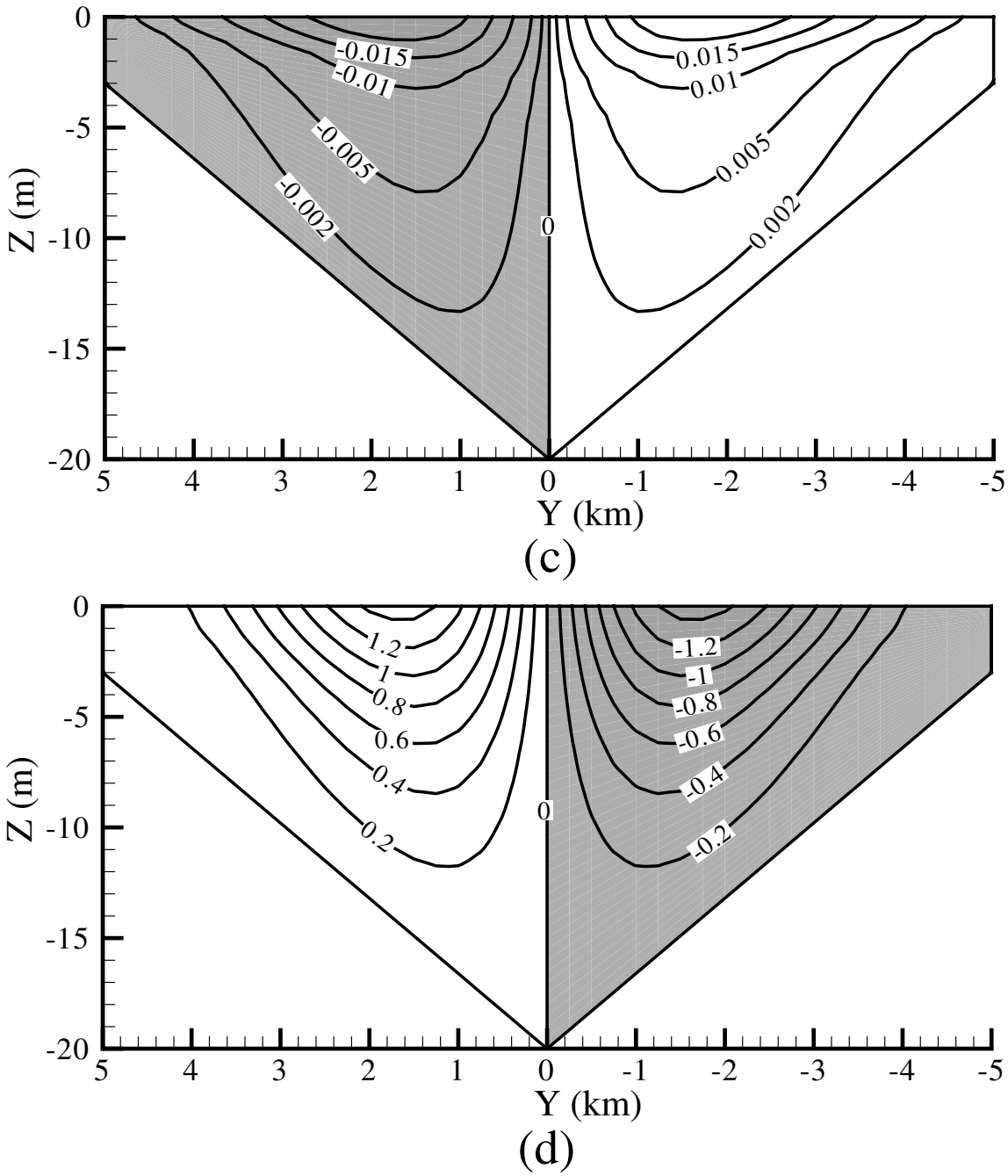


Figure 7.11: Contours of u_y velocity $\left(m/s \times 10^{-2}\right)$ at (c) $x = 50\,km$, and (d) $x = 98\,km$ for a non-rotating system with the triangular bathymetry profile. The dark areas represent negative velocities.

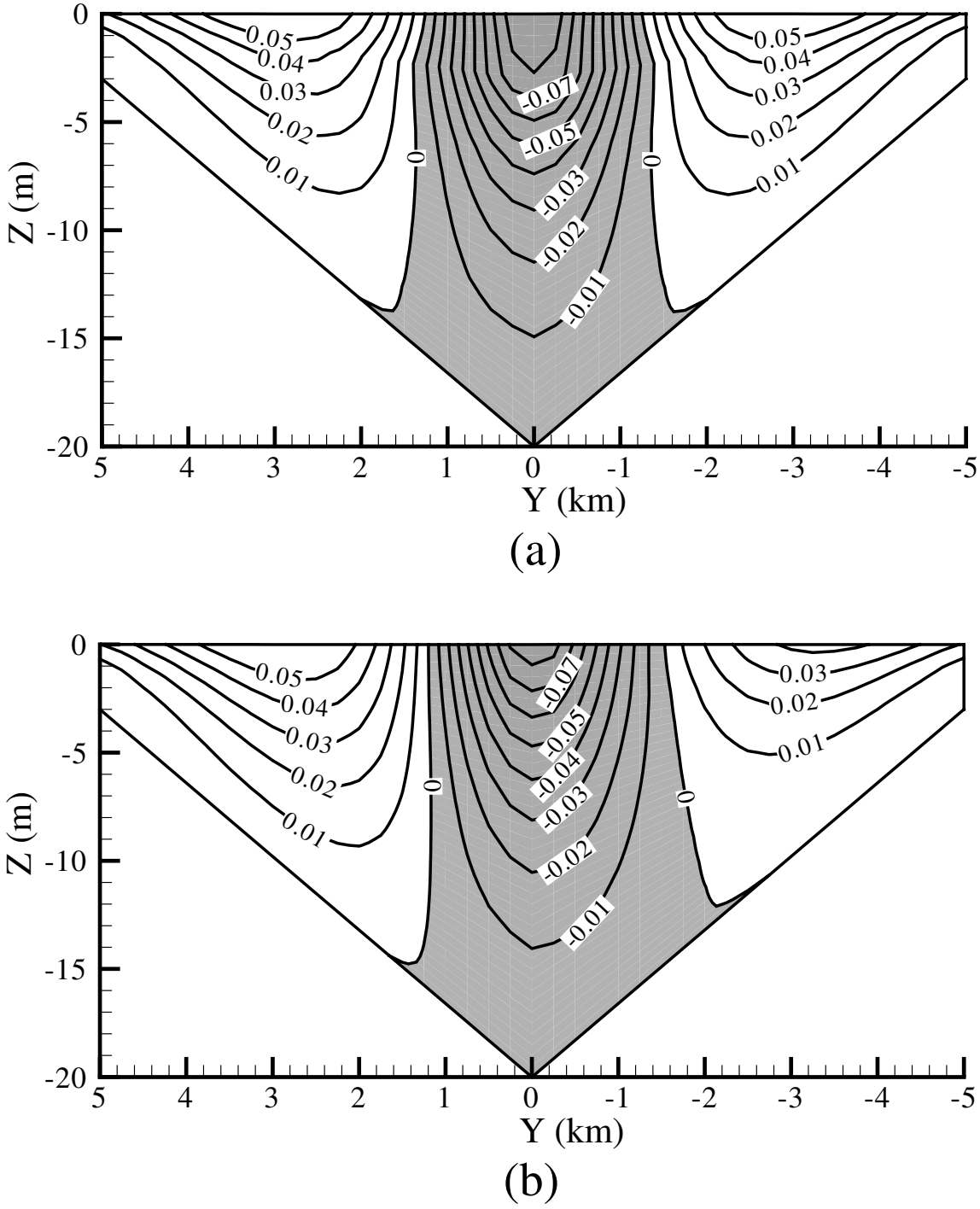


Figure 7.12: Contours of the u_x velocity (m/s) at (a) $x = 50\ km$, and (b) $x = 98\ km$ for a rotating system with the triangular bathymetry profile. The dark areas represent negative velocities.

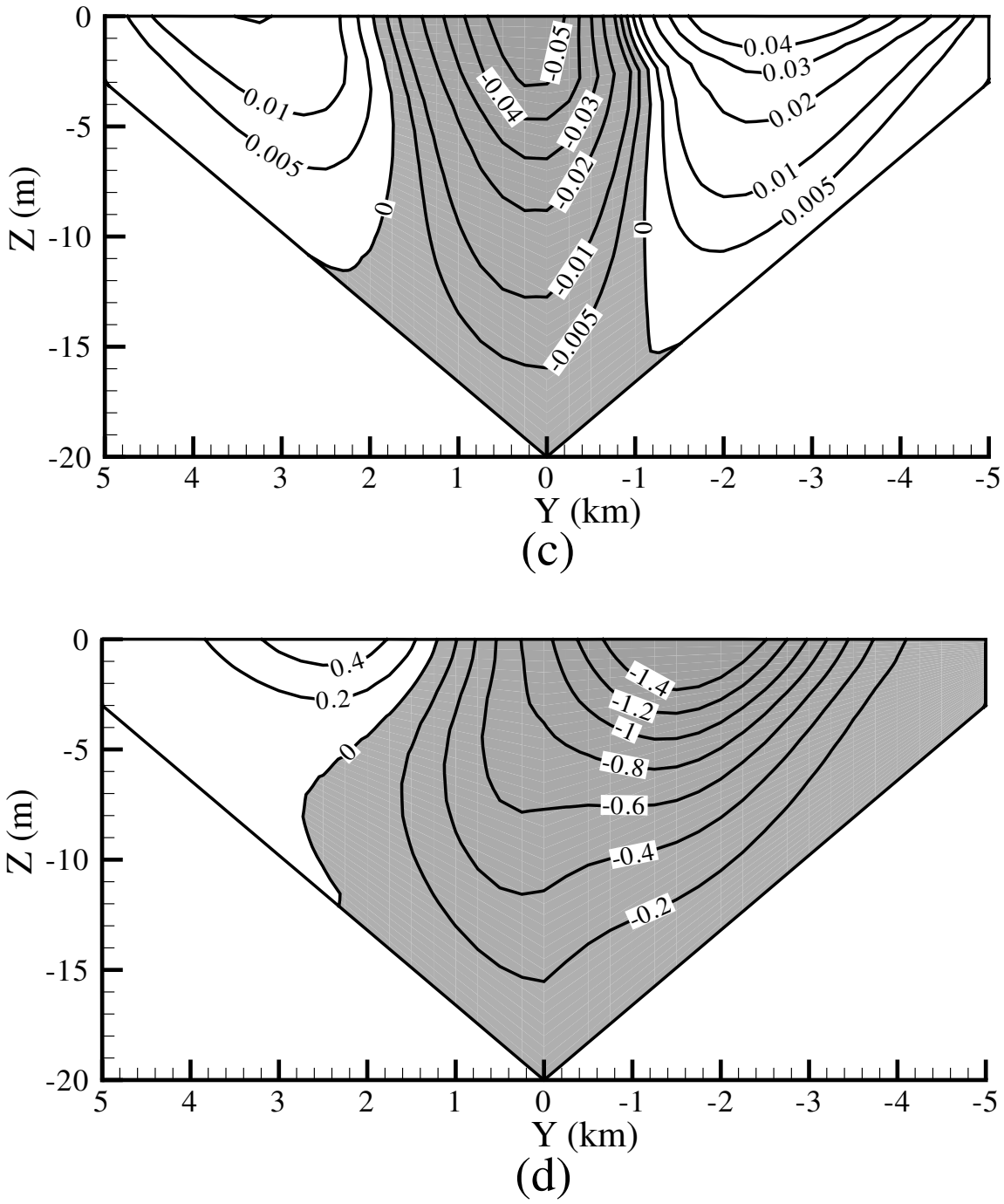


Figure 7.12: Contours of the u_y velocity $\left(m/s \times 10^{-2}\right)$ at (c) $x = 50\,km$, and (d) $x = 98\,km$ for a rotating system with the triangular bathymetry profile. The dark areas represent negative velocities.

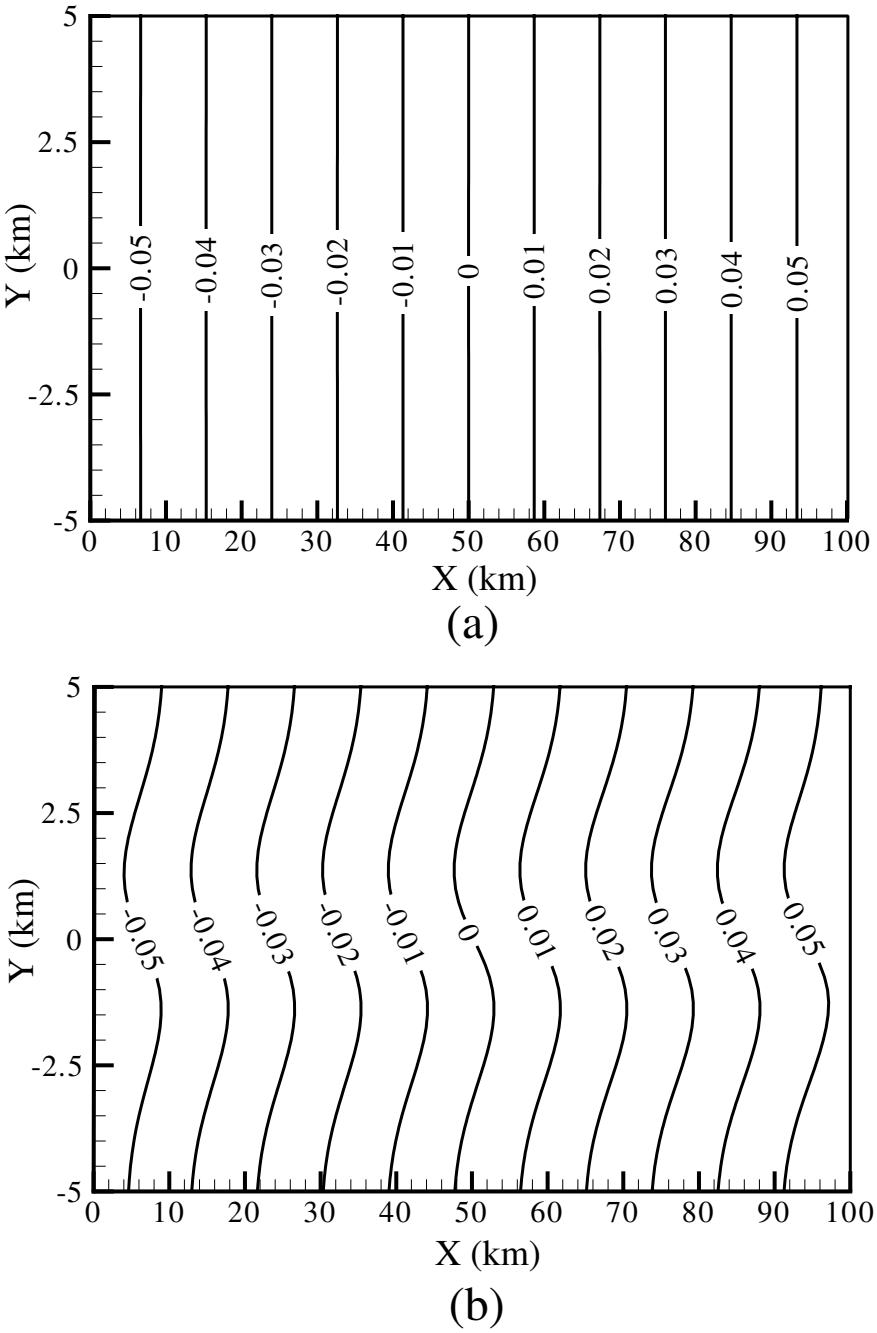


Figure 7.13: Contours of the water surface elevation (m) for (a) non-rotating, and (b) rotating cases for the triangular bathymetry profile.

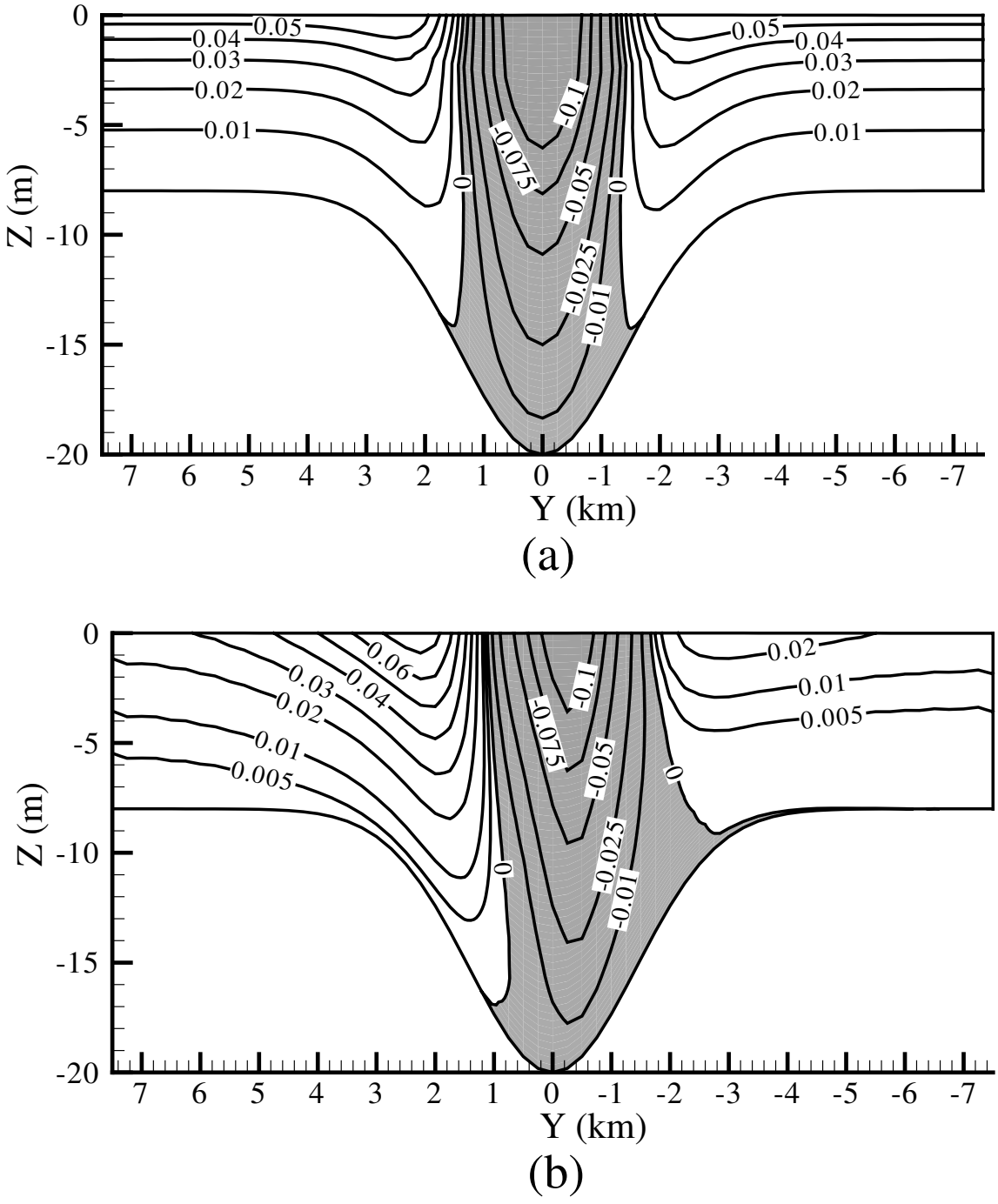


Figure 7.14: Contours of u_x velocity (m/s) at (a) $x = 50\,km$, and (b) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.2). The dark areas represent negative velocities.

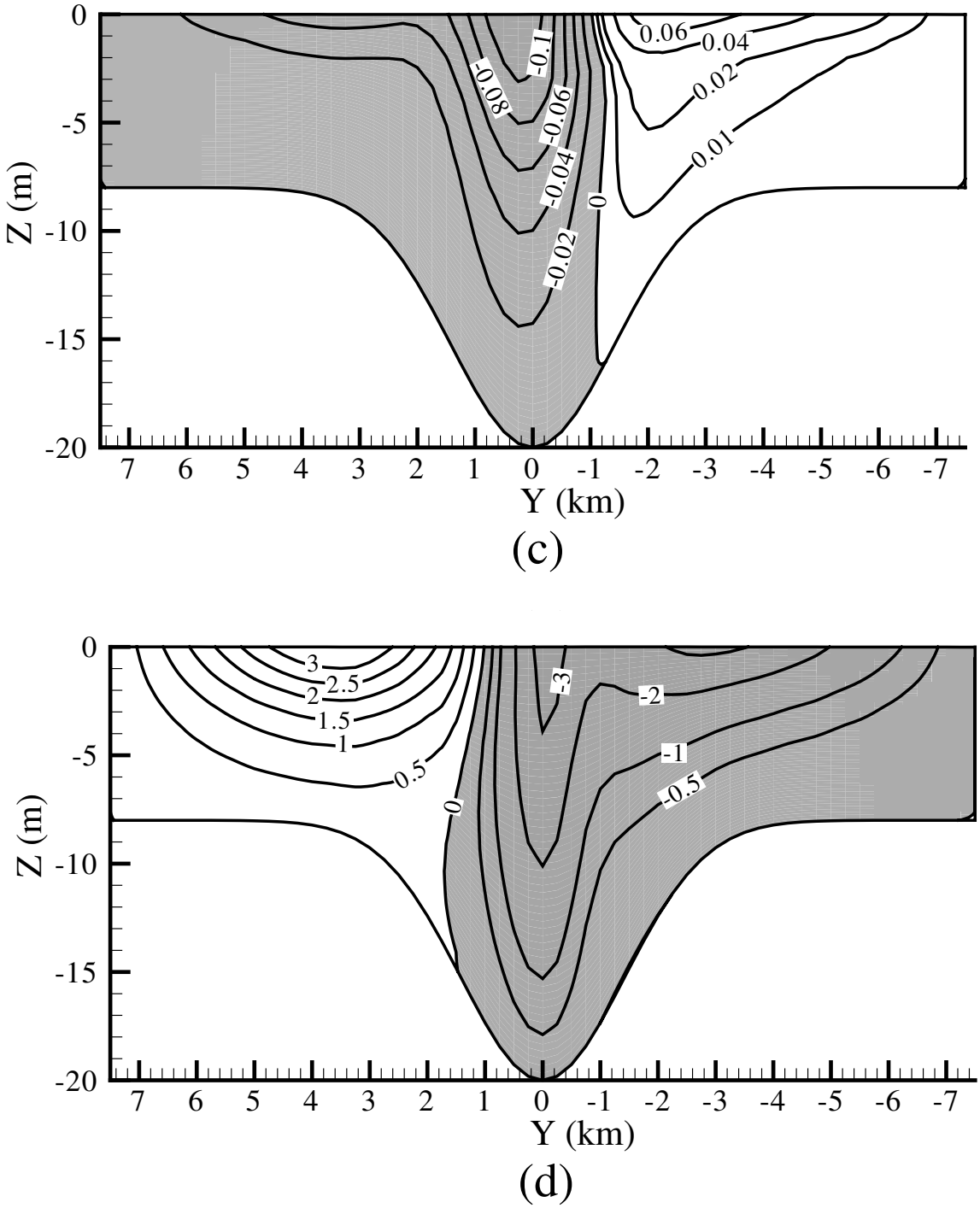


Figure 7.14: Contours of u_y velocity $\left(m/s\times10^{-2}\right)$ at (c) $x=50\,km$, and (d) $x=98\,km$ for the Gaussian bathymetry profile given in equation (7.1.2). The dark areas represent negative velocities.

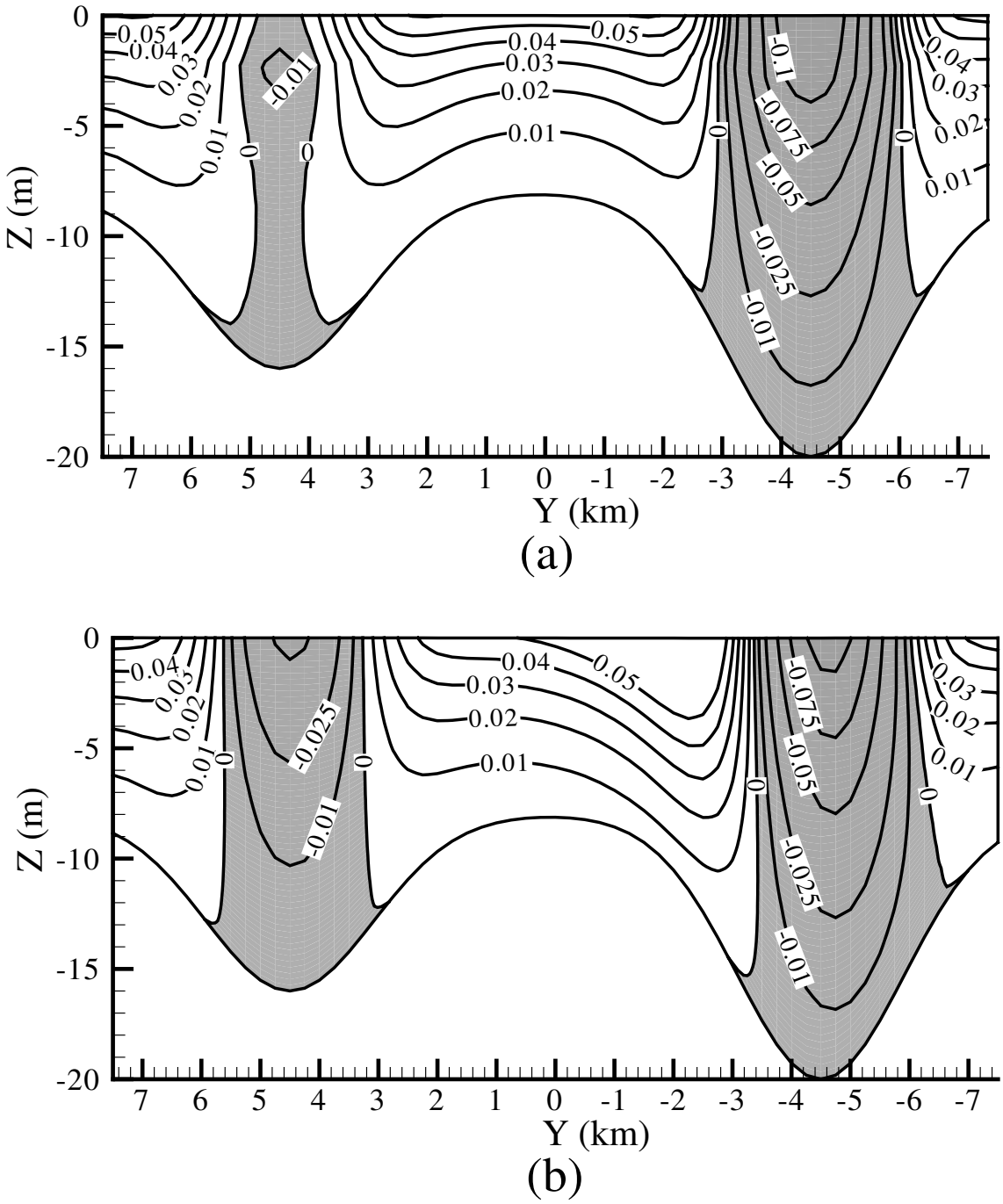


Figure 7.15: Contours of u_x velocity (m/s) at (a) $x = 50\,km$, and (b) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.

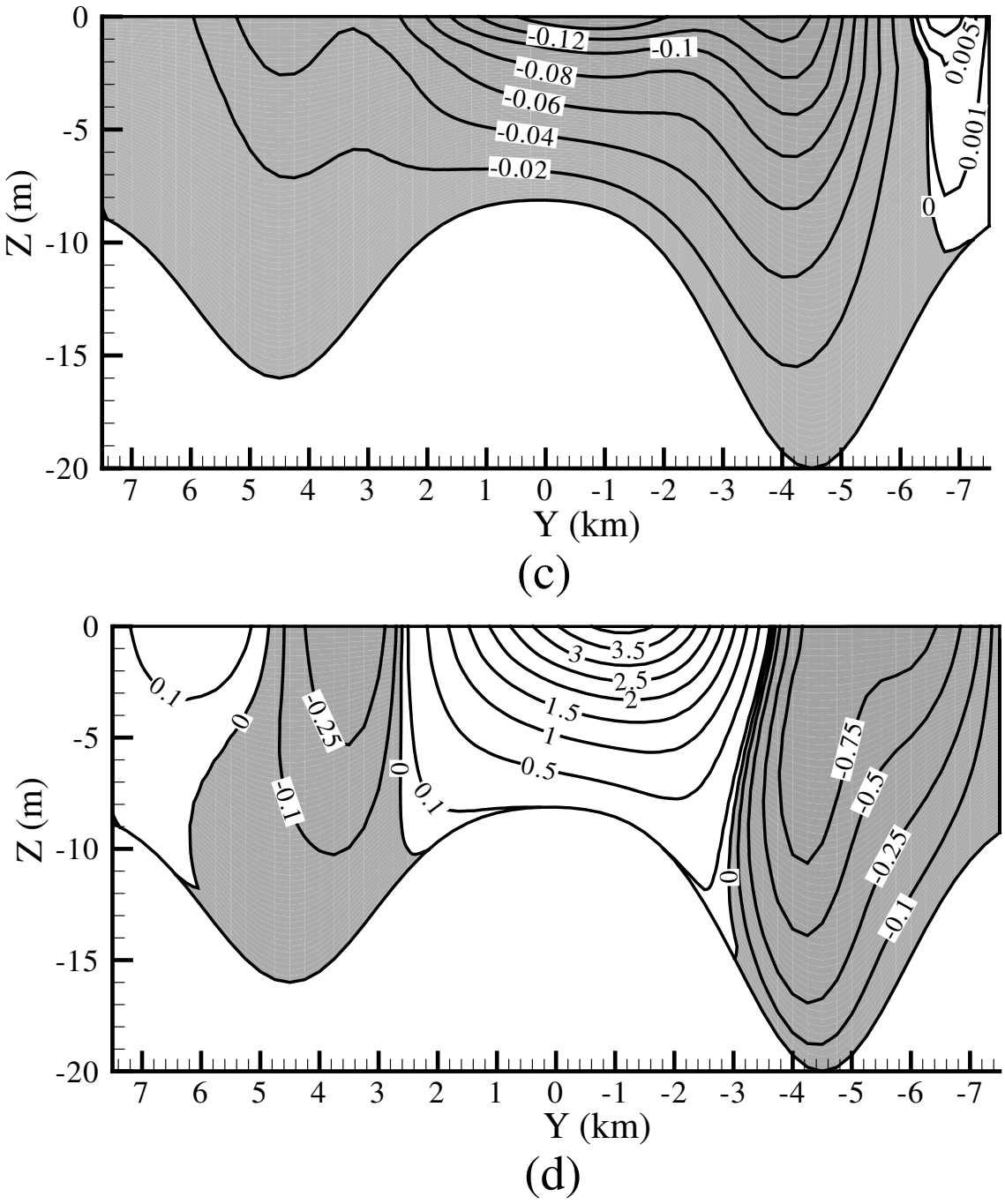


Figure 7.15: Contours of u_y velocity $\left(m/s\times 10^{-2}\right)$ at (c) $x = 50\ km$, and (d) $x = 98\ km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.

LBM is calculated using equation (3.3.13): $\tau - 0.5 = 1.2\times10^{-9}$. The relaxation rates $s_4 = s_6 = s_7 = s_8 = 1/\tau$, and $s_1 = s_2 = s_7 - 0.6$ were used. The free-slip condition was used for all closed boundaries. The initial water in the basin was static, the horizontal density gradient was constant, and a wind stress was increased linearly during the first six simulated hours. After six hours, the wind was constant. The wind stress acted along the positive x direction and blew uniformly throughout the domain. The numerical simulations were run up to 2 days after the wind stress was constant with the establishment of a steady state velocity field occurring after about 1 day. The numerical simulations were run in MATLAB on a single workstation with a 3.0 GHz Intel® Core™2 Extreme quad core processor and an NVIDIA® Tesla™ C1060 Computing Processor. The parallel performance of a single core of the quad core processor and the Tesla are compared.

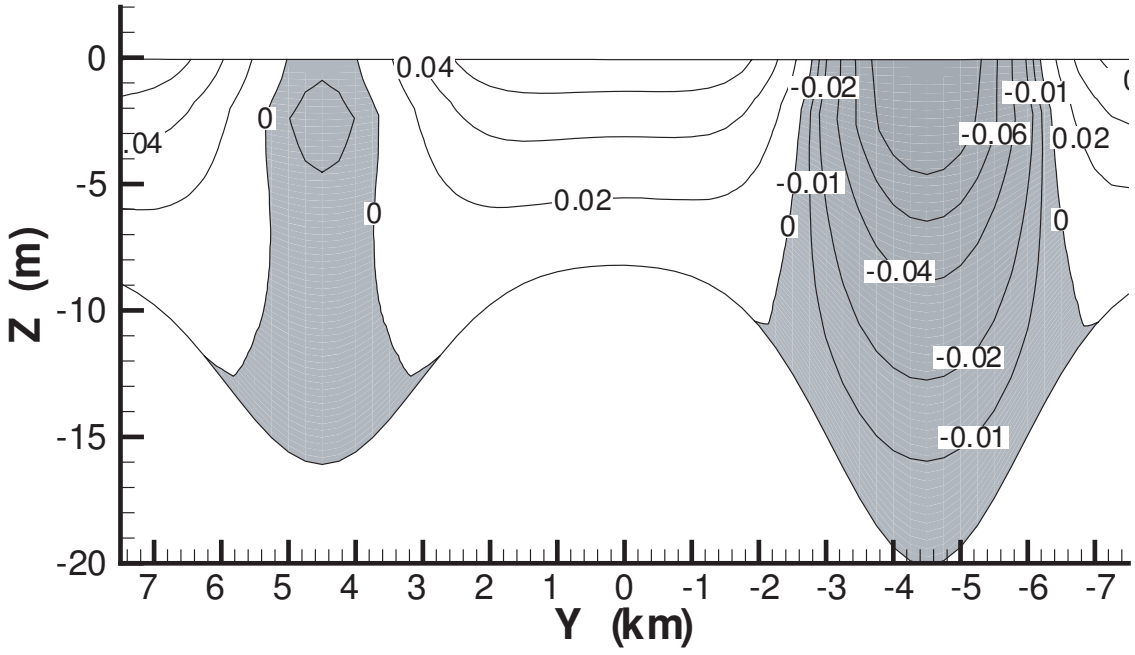
For the wind-driven case, the physical parameters are $\tau_{iz}^{W} = 0.04\ N/m^2$, $\partial\rho/\partial x = 0$, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.0025\ m/s$, and $f_c = 1\times10^{-4}\ s^{-1}$. The wind stress was applied in the positive x direction. The wind velocity is $U_{Wx} = 4.0825\ m/s$ and $U_{Wy} = 0\ m/s$. The bed friction is formulated using a linear friction law. Figure 7.16 shows the u_x and u_y distributions at $x = 50\ km$ and $x = 98\ km$ planes. Similar to previous results, the u_x flows in the direction of the wind at all depths in the shallow regions along the transverse boundaries and center of the channel as shown in Figures 7.16a and 7.16b,. The u_x flows in the opposite direction of the wind in the deep parts of the channel. Figures 7.16c and 7.16d show the contours of the transverse flow, u_y at $x = 50\ km$ and $x = 98\ km$ planes.

For the density-driven case, the physical parameters are $\tau_{iz}^{W} = 0$, $\partial\rho/\partial x = -5\times10^{-8}\ kg/m$, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.005\ m/s$, and $f_c = 1\times10^{-4}\ s^{-1}$. Figure 7.17 shows the u_x and u_y distributions at $x = 50\ km$ and $x = 98\ km$ planes. In the case of the density-driven flow, the u_x flows in the direction of the horizontal gradient at all depths in the shallow regions along the transverse boundaries and center of the channel in Figures 7.17a and 7.17b,. The u_x flows in the opposite direction of the horizontal gradient in the deep parts of the channel. These flow features are the opposite of those found for the pure wind driven case. The magnitude of the flow is highest near the surface and decreases with depth as expected from the bottom friction. Figures 7.17c and 7.17d show the contours of the transverse flow, u_y at $x = 50\ km$ and $x = 98\ km$ planes. Although the flow field is reversed for this case, the effect of the Earth's rotation is consistent with the earlier examples producing surface elevation contours with strong lateral variability, which is to be expected. The transverse velocities exhibit similar behavior as the wind driven case with stronger magnitude near the $x = 98\ km$ plane. However, along the $x = 50\ km$ plane, the velocities are small yet exhibit a vertical distribution of positive and negative flow.

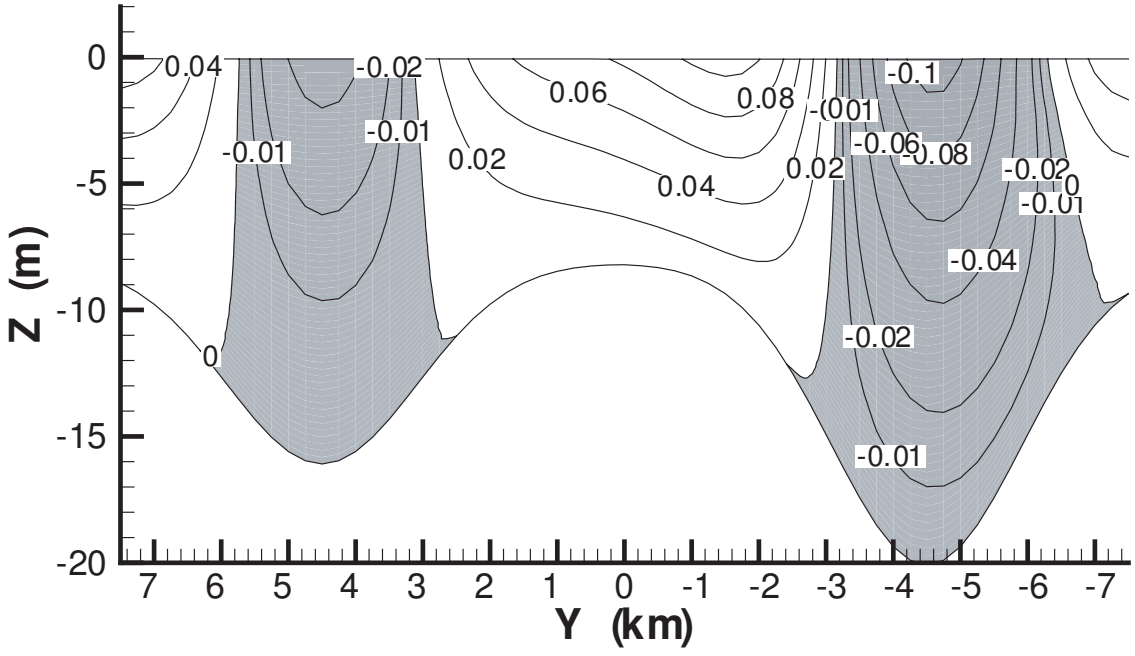
For the combined wind- and density-driven case, the physical parameters are $\tau_{iz}^{W} = 0.04\ N/m^2$, $\partial\rho/\partial x = -5\times10^{-8}\ kg/m$, $\partial\rho/\partial y = 0$, $\rho = 1025\ kg/m^3$, $\rho_a = 1.2\ kg/m^3$, $C_W = 0.0015$, $\mu = 0.004\ cm^2/s$, $\kappa = 0.005\ m/s$, and $f_c = 1\times10^{-4}\ s^{-1}$. The wind stress was applied in the positive x direction. Figure 7.18 shows the u_x and u_y distributions at $x = 50\ km$ and $x = 98\ km$ planes. For the combined of wind- and density-driven case, the u_x distribution is similar to the density driven case in terms of direction of the flow in shallow and deep regions as shown in Figures 7.18a and 7.18b. Figures 7.18c and 7.18d show the contours of the transverse

flow, u_y at $x = 50\,km$ and $x = 98\,km$ planes. The flow features with respect to bottom friction, Earth's rotation, and bathymetry are all consistent with previous results. The main difference in the combined case is that the magnitude of the flow is smallest near bed, increases in the positive z-direction then decreases again near the surface. This is expected due to the bottom friction and the wind stress occurring in the opposite direction of the density-gradient. The density gradient accounts for the direction of the flow, while the wind stress accounts for the smaller magnitude velocities near the surface.

The simulation time for each case was 47.3 hours on a single core of the CPU and 1.68 hours on the Tesla GPU demonstrating 28.16 times speedup.

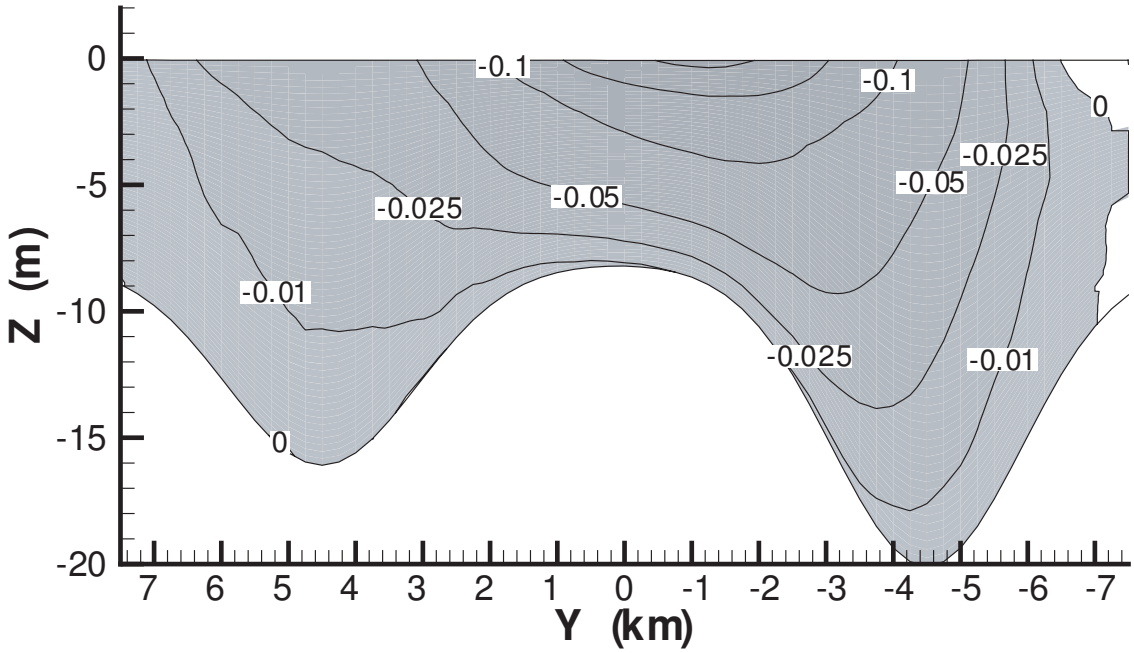


(a)

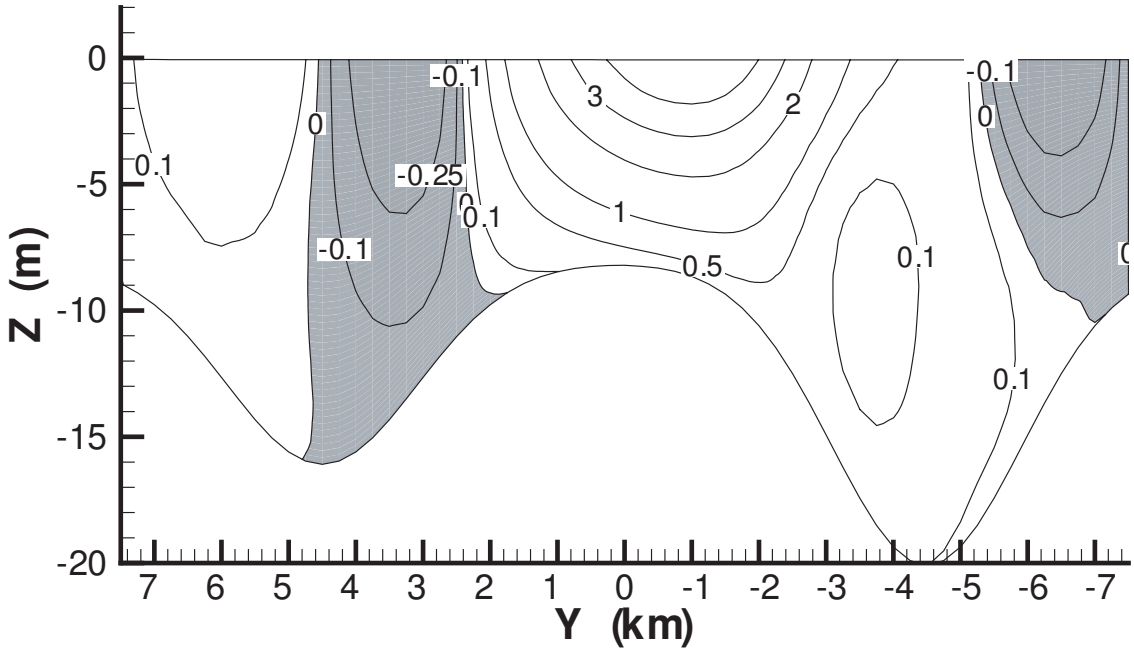


(b)

Figure 7.16: Contours of u_x velocity (m/s) at (a) $x = 50\,km$, and (b) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities

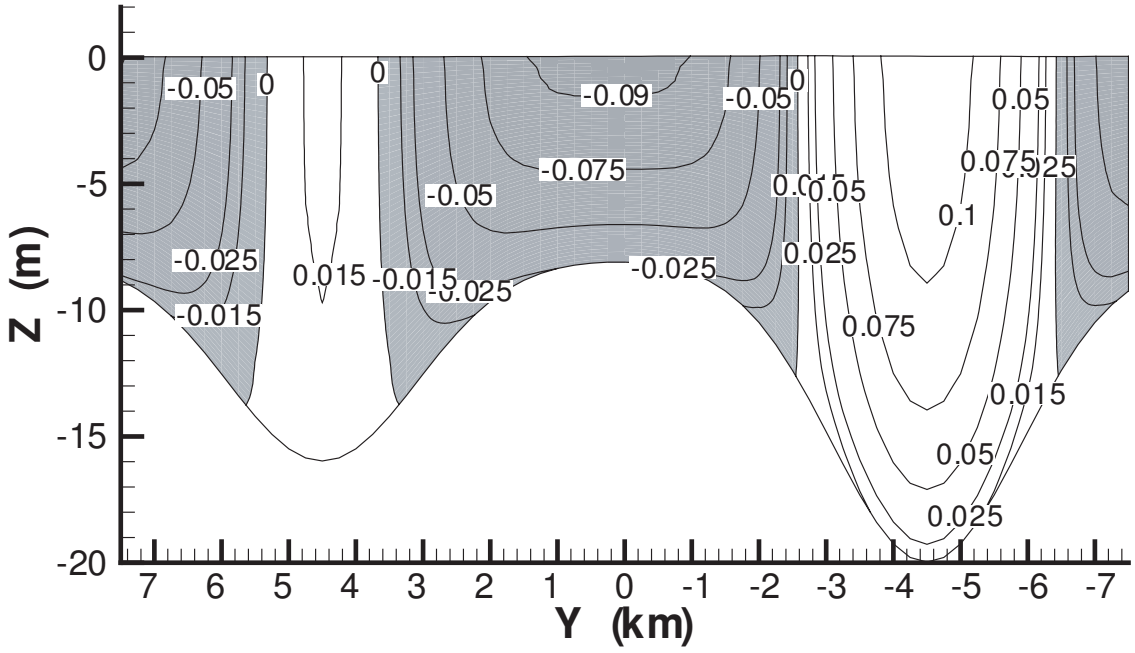


(c)

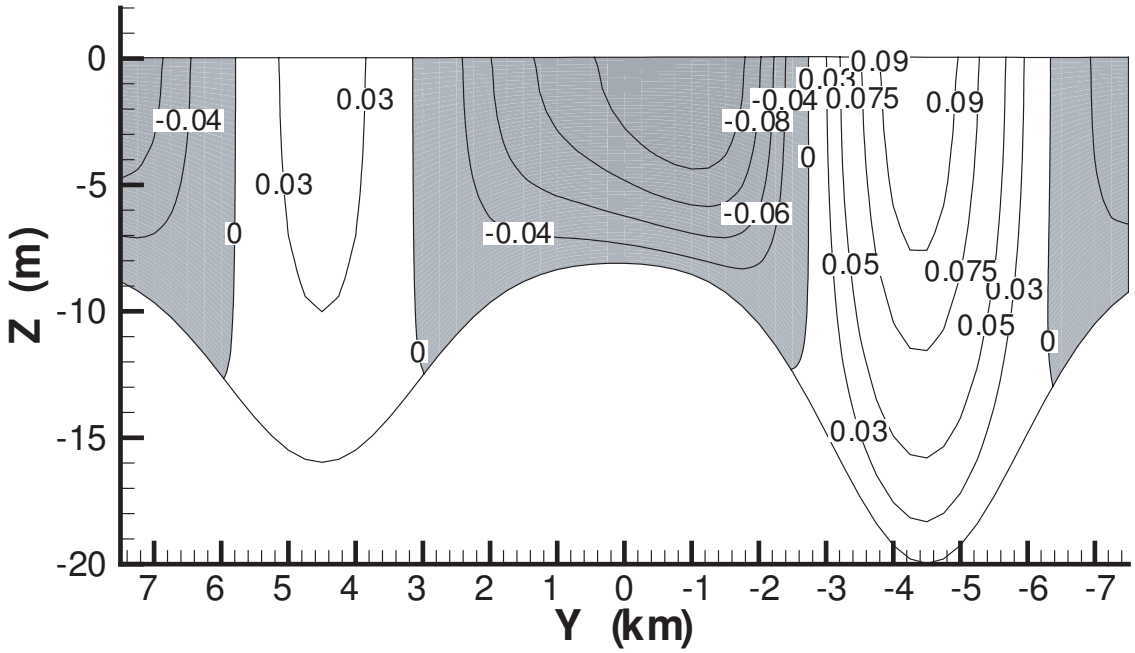


(d)

Figure 7.16: Contours of u_y velocity $\left(m/s\times10^{-2}\right)$ at (c) $x=50\,km$, and (d) $x=98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.

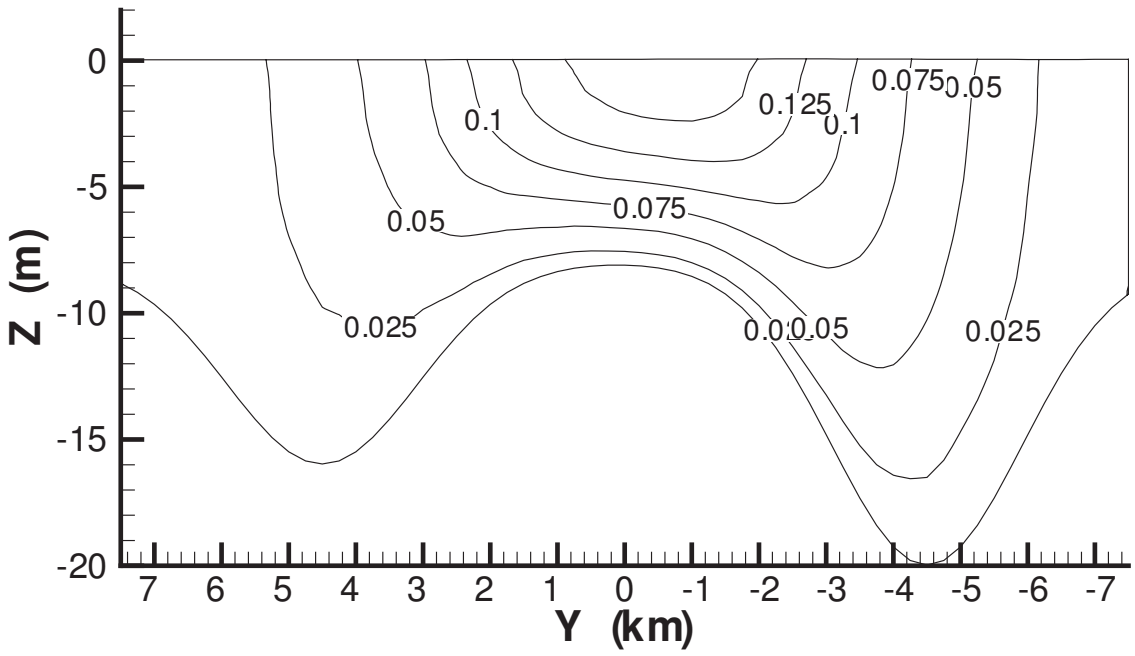


(a)

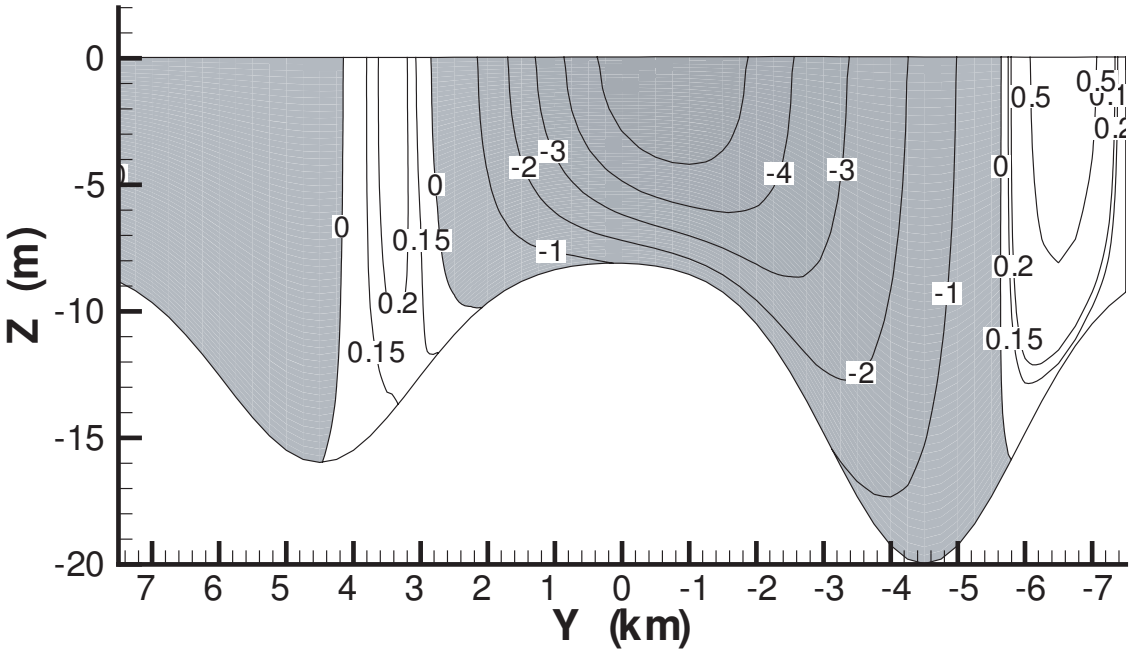


(b)

Figure 7.17: Contours of u_x velocity (m/s) at (a) $x = 50\,km$, and (b) for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.

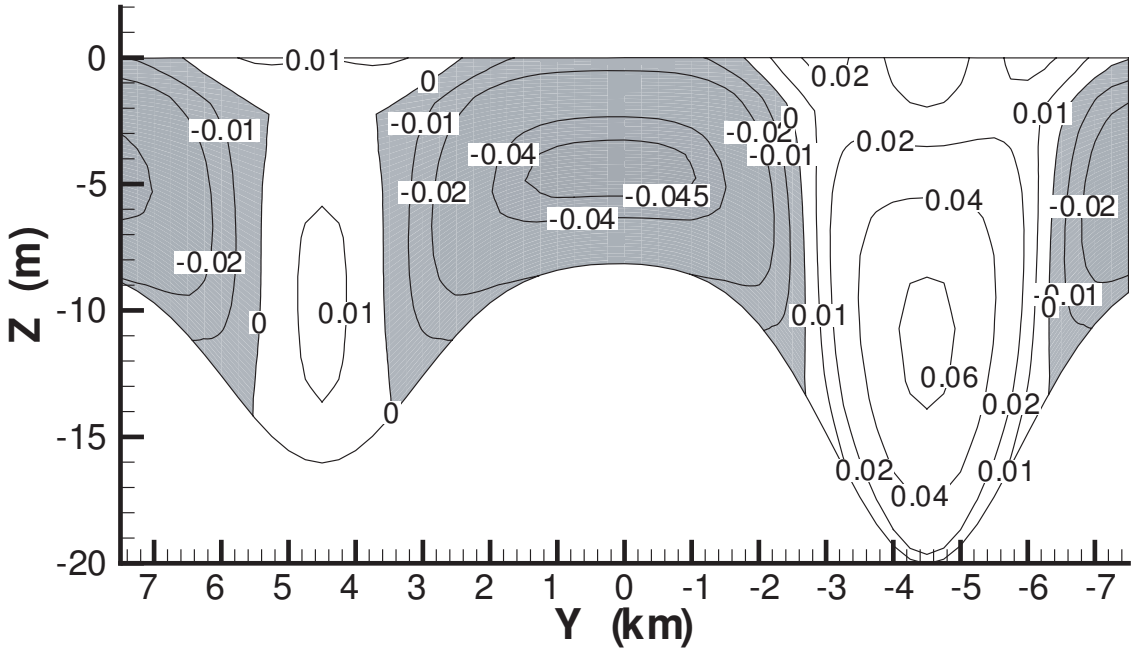


(c)

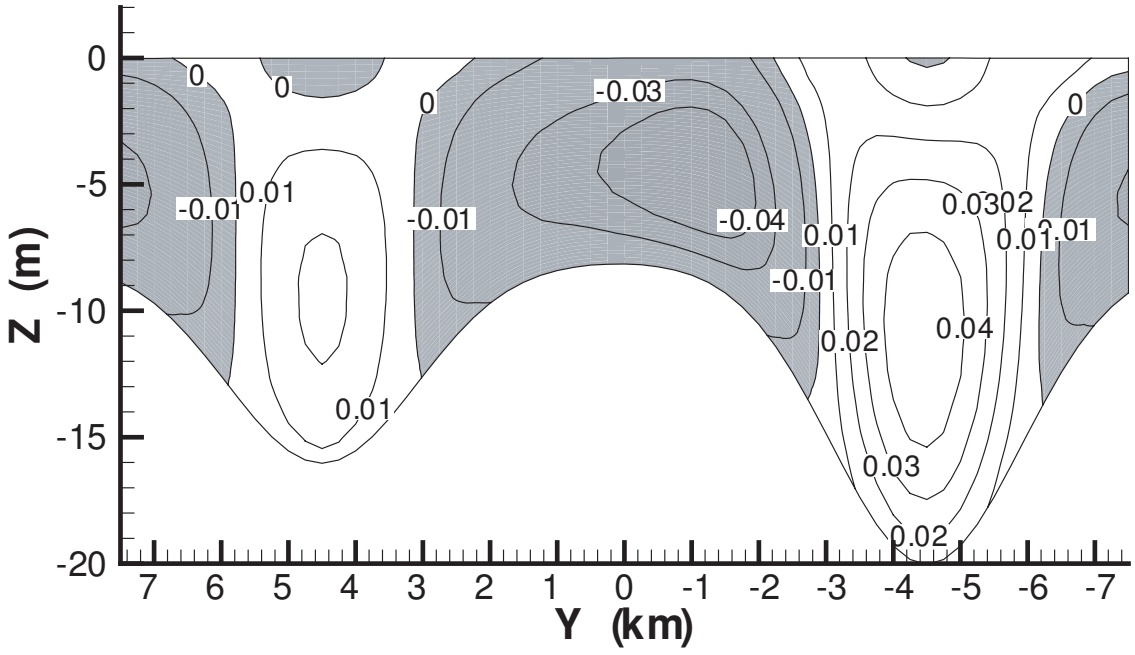


(d)

Figure 7.17: Contours of u_y velocity $\left(m/s \times 10^{-2}\right)$ at (c) $x = 50\,km$, and (d) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.

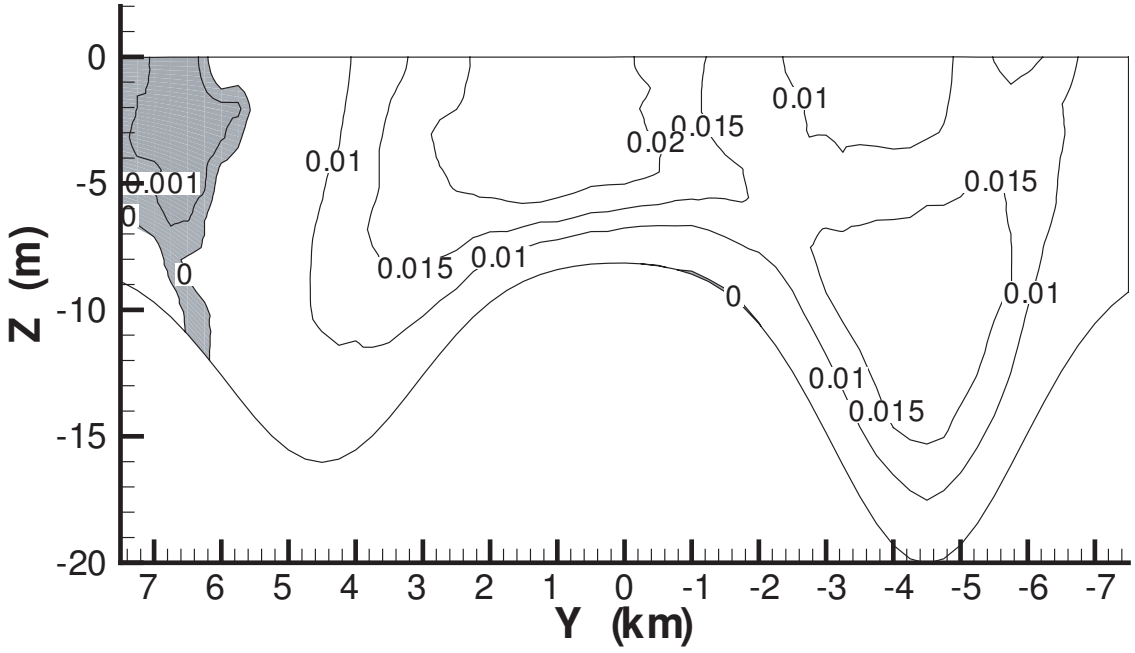


(a)

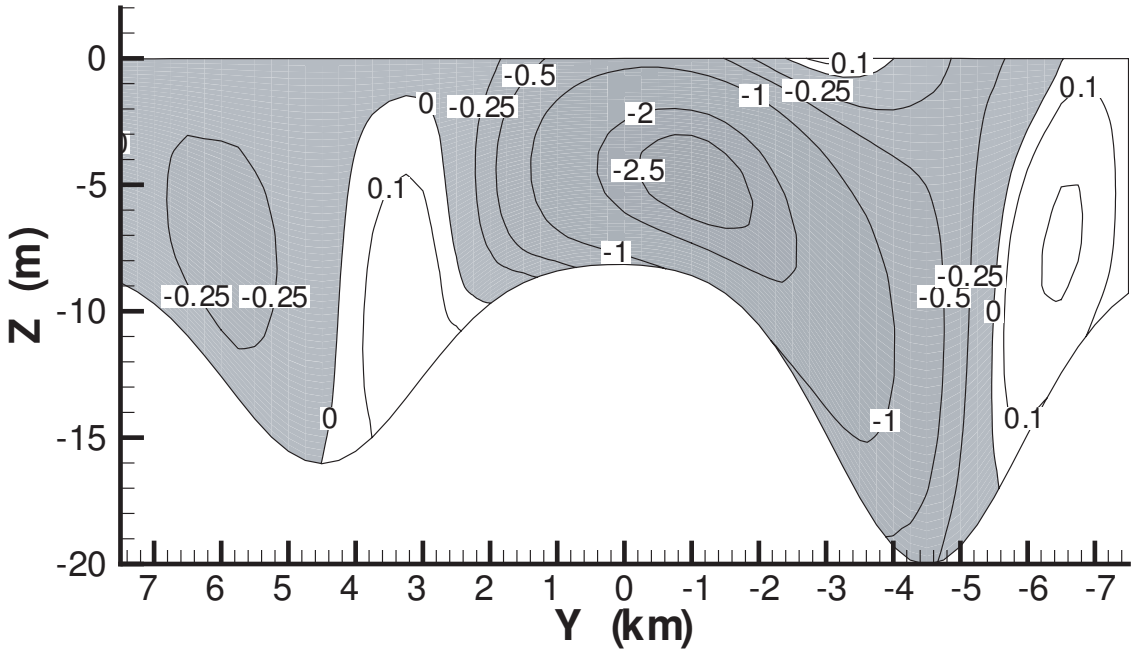


(b)

Figure 7.18: Contours of u_x velocity (m/s) at (a) $x = 50\,km$, and (b) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities.



(c)



(d)

Figure 7.18: Contours of u_y velocity $\left(m/s\times 10^{-2}\right)$ at (c) $x = 50\,km$, and (d) $x = 98\,km$ for the Gaussian bathymetry profile given in equation (7.1.3). The dark areas represent negative velocities

8 CONCLUSIONS

Coastal wetland restoration and management is an area of high impact socially, economically and ecologically. Understanding the physical processes involved as well as understanding and managing how human activities affect these processes is of great importance. Numerical modelling and simulation serves as a tool to both validate mathematical models used to understand these physical processes and how they interact and predict and analyze the affects of human activities. Ultimately, improved modelling capabilities and the ability to simulate multiple scenarios while studying how they interact with each other leads to more sophisticated decision making and management tools. This dissertation focused on study, validation, and demonstration of the lattice Boltzmann method as a numerical modelling and simulation tool for two- and three- dimensional flows in the shallow water regime in high performance computing environments. The concluding remarks for the dissertation are the following:

For the two-dimensional shallow water equations, the MRT collision operator out performs the SRT (BGK) collision operator for the shallow water equations at the situation that the relaxation time parameter, τ, is close to the stability limit of 0.5. The MRT collision operator was able to increase stability and accuracy and eliminate spurious oscillations when the SRT model fails. The dam break problem demonstrates that the MRT-LBM is able to handle complex flow structures such as shocks, rarefaction waves and contact discontinuities. For the two-dimensional anisotropic advection dispersion equation, the TRT-LBM with speed-of-sound techniques was able to account for the heterogeneity and anisotropy in the velocity dependent dispersion coefficient. Specifically, the speed-of-sound techniques are able to cope with the discontinuous free surface water depth in the transport problem. Excellent agreement is obtained between numerical predictions and analytical solutions for both the hydrodynamic and transport

equations. The combination of MRT-LBM and TRT-LBM to predict mass transport with velocity-dependent dispersion in shallow water flow due to partial dam breaks is demonstrated with the numerical results indicating that the present method is promising for modeling transport phenomena in shallow water flows

The two-dimensional LBM for shallow water equation has been extended to solve three-dimensional wind-driven and density-driven circulation by introducing a multi-layer LB model. The advantage of the multi-layer LB model is that it avoids the computationally expensive solution of the Navier-Stokes equations and obtains stratified horizontal flow velocities as vertical velocities are relatively small and the flow is still within the shallow water regime. A MRT-LBM model is used to solve for each layer coupled by the vertical viscosity forcing term. To increase solution stability, an implicit step is chosen to obtain stratified velocities distributions. The main advantage of using the LBM is that after selecting appropriate EDFs, the LB algorithm is only slightly modified for each layer and retains all the simplicities in the LBM within the high performance computing (HPC) environment.

The multi-layer LB model has been implemented to simulation three dimensional wind- and density-driven circulation. The influence of wind stress, horizontal density gradient, vertical viscosity forcing, bottom friction and bathymetry were tested. The numerical results of flow velocities for wind-driven circulation in a rectangular lake with a flat bottom agree well with the analytical solutions (Shankar et al. 1997). Moreover, the multi-layer LB model was tested over non-uniform bathymetry with and without the effects of the Earth's rotation to calculate lateral and vertical distributions of the velocities. The simulated wind-driven circulation patterns, consisting of downwind flow over the shallow regions and upwind flow in the deep region along the entire domain, were consistent with other studies (Glorioso and Davies 1995; Sanay and

Valle-Levinson 2005). The simulated density-driven and combined wind- and density-driven circulation patterns were consistent with the wind-driven results.

The parallel performance of the shallow water LBM and multi-layer LB model has been investigated on central processing unit (CPU) based and graphics processing unit (GPU) based high performance computing (HPC) architectures. A key point to understand in the implementation of the parallel LBM is that the two step solution procedure, (stream and collide), involves only a data shift to nearest neighbor nodes and local computations. This inherent parallelism is efficient on both CPU based on GPU based architectures; however, the performance must be optimized for each architecture. For the two dimensional LBM, the parallel domain decomposition and data access patterns should be selected to take advantage of cache optimization using explicit loop control on the CPU and shared memory optimization on the GPU. The parallel performance on the CPU was improved with the explicit loop control and depends on the block size chosen. The parallel performance on the GPU increases with problem size due to increasing computational intensity and decreasing need for communication sub-domains of the data. The parallel performance of the multi-layer LB model shared the same characteristics on the CPU and GPU; however the parallel decomposition along only on the horizontal flow directions has two advantages: 1.) It retains the inherent parallelism of the LBM for each layer; and 2.) It retains the locality of the tridiagonal solver over layers with respect to threads.

Future Work

The work presented in this dissertation provides a foundation for an interdisciplinary study focused on improving numerical modeling and simulation tools for coastal modeling, restoration and management. The aim of this dissertation will be to extend the capabilities of the LBM for

coastal modeling and implement the LB models in a computational framework capable of taking advantage of heterogeneous HPC systems. The future work can include:

1. Performing a higher order recovery of the multi-layer shallow water equations using the MRT collision operator in order to perform a stability analysis to provide a systematic basis for choosing MRT parameters.
2. Implementing and characterizing the parallel performance of the parallel two-dimensional and multi-layer LB models in a high level language such as C using hybrid shared memory (OpenMP), distributed memory (MPI) techniques for CPU-based HPC systems.
3. Implementing and characterizing the parallel performance of the parallel two-dimensional and multi-layer LB modes in a high level language such as Open Computing Language (OpenCL) for GPU-based HPC systems.

Performing the higher order Chapman Enskog recovery of the multi-layer shallow water equations and stability analysis will improve the understanding of the MRT collision operator while providing a guide for parameter selection. Implementing the parallel LB solvers using hybrid OpenMP / MPI techniques on the CPU and OpenCL on the GPU will allow the parallel features of the algorithm to be well understood leading to efficient implementation on different architectures. The future of high performance computing will rely on a combination of distributed and shared memory systems based on traditional CPU-based architectures as well as accelerators such as GPU-based architectures. The over arching theme is that the parallel features of the algorithm must be well studied and designed to be flexible enough for heterogeneous HPC systems. The research into the hybrid OpenMP / MPI implementations on the CPU and OpenCL implementation on the GPU will provide the foundation for LB solver that is flexible and

efficient on various heterogeneous HPC systems. The outcome of the project is a more mature parallel LB solver for two and three dimensional shallow water modeling providing a better tool to research computational techniques for computational efficiency and coupling between various models on heterogeneous HPC systems. The parallel LB solver can then be used to study the natural processes affecting coastal wetland loss running scenarios in the Lake Ponchartrain Basin.

REFERENCES

Abbott, M. and A. Minns (1998). Computational Hydraulics, Ashgate Publishing.

Accelereyes, L. (2008). "MATLAB GPU Computing." from http://www.accelereyes.com/.

Accelereyes, L. (2009). "Accelereyes Jacket User Guide Version 1.2.1." Retrieved 01 December 2009, from http://www.accelereyes.com/doc/JacketUserGuide.pdf.

Aizinger, V. and C. Dawson (2002). "A discontinuous Galerkin method for two-dimensional flow and transport in shallow water." Advances in Water Resources **25**(1): 67-84

Al-Barwani, H. H. and A. Purnama (2008). "Simulating brine plumes discharged into the seawaters." Desalination **221**(1-3): 608-613.

Alcrudo, F. and F. Benkhaldoun (2001). "Exact solutions to the Riemann problem of the shallow water equations with a bottom step." Computers & Fluids **30**(6): 643-671.

Alcrudo, F. and P. Garcia-Navarro (1993). A high-resolution Godunov-type scheme in finite volumes for the 2D shallow-water equations. **16:** 489-505.

Alexander, F. J., H. Chen, S. Chen and G. D. Doolen (1992). "Lattice Boltzmann Model for Compressible Fluids." Physical Review A **46**(4): 1967-1970.

Ambrosi, D. (1995). "Approximation of shallow water equations by Roe's Riemann solver." International Journal for Numerical Methods in Fluids **20**(2): 157-168.

Audusse, E. (2005). "A multilayer Sainte-Venant model." Discrete and Continuous Dynamical Systems, Series B **5**(2): 189-214.

Audusse, E. and M.-O. Bristeau (2007). "Finite-Volume Solvers for a Multilayer Saint-Venant System." International Journal of Applied Mathematics and Computer Science **17**(3): 311-320.

Audusse, E., M. O. Bristeau and A. Decoene (2006). 3D Free Surface Flows Simulations Using a Multilayer Saint-Venant Model. Comparisons with Navier-Stokes Solutions. Numerical Mathematics and Advanced Applications**:** 181-189.

Audusse, E., M. O. Bristeau and A. Decoene (2008). "Numerical simulations of 3D free surface flows by a multilayer Saint-Venant model." International Journal for Numerical Methods in Fluids **56**(3): 331-350.

Banda, M. K. and M. S. G. Thömmes (2009). "Lattice Boltzmann simulation of dispersion in two-dimensional tidal flows." International Journal for Numerical Methods in Engineering **77**(6): 878-900.

Bella, G., S. Filippone, N. Rossi and S. Ubertini (2002). Using openMP on a hydrodynamic lattice-Boltzmann code. Fourth European Workshop on OpenMP, Roma.

Benkhaldoun, F., I. Elmahi and M. SeaÏ¨d (2007). "Well-balanced finite volume schemes for pollutant transport by shallow water equations on unstructured meshes." Journal of Computational Physics **226**(1): 180-203.

Bernsdorf, J., F. Durst and M. Schäfer (1999). "Comparison of cellular automata and finite volume techniques for simulation of incompressible flows in complex geometries." International Journal for Numerical Methods in Fluids **29**(3): 251-264.

Bhatnagar, P. L., E. P. Gross and M. Krook (1954). "A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems." Physical Review **94**(3): 511-525.

Blumberg, A. F. and G. L. Mellor (1987). A description of a three-dimenstional coastal ocean circulation model. Three-Dimensional Coastal Ocean Models. N. S. Heaps. Washington, DC, American Geophysical Union.

Board, O. S. (2006). Drawing Louisiana's New Map: Addressing Land Loss in Coastal Louisiana, National Academies Press.

Board, O. A. R. (2008). 2008, from http://www.openMP.org.

Bolz, J., I. Farmer, E. Grinspun and P. Schroder (2003). "Sparse matrix solvers on the GPU: conjugate gradients and multigrid." ACM Trans. Graph. **22**(3): 917-924.

Breuer, M., J. Bernsdorf, T. Zeiser and F. Durst (2000). "Accurate computations of the laminar flow past a square cylinder based on two different methods: Lattice-Boltzmann and finite-volume." International Journal of Heat and Fluid Flow **21**(2): 186-96.

Buck, I. (2005). Stream computing on graphics hardware, Stanford University**:** 117.

Cai, L., W.-X. Xie, J.-H. Feng and J. Zhou (2007). "Computations of transport of pollutant in shallow water." Applied Mathematical Modelling **31**(3): 490-498.

Cao, Z., G. Pender, S. Wallis and P. Carling (2004). "Computational Dam-Break Hydraulics over Erodible Sediment Bed." Journal of Hydraulic Engineering **130**(7): 689-703.

Casulli, V. and R. Walters (2000). "An unstructured grid, three-dimensional model based on the shallow water equations." International Journal for Numerical Methods in Fluids **32**(3): 331-348.

Chaudry, M. H. (1993). Open Channel Flow. Englewood Cliffs, New Jersey, Prentice Hall.

Chen, H., S. Chen and W. H. Matthaeus (1992). "Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method." Physical Review A **45**(8): R5339.

Chen, S., S. P. Dawson, G. D. Doolen, D. R. Janecky and A. Lawniczak (1993). Lattice Methods and Their Applications To Reacting Systems.

Chen, S. and G. D. Doolen (1998). "Lattice Boltzmann method for fluid flows." Annual Review of Fluid Mechanics **30**: 329-364.

Chen, S. Y., Z. Wang, X. W. Shan and G. D. Doolen (1992). "Lattice Boltzmann Computational Fluid-Dynamics in 3 Dimensions." Journal of Statistical Physics **68**(3-4): 379-400.

Chertock, A., A. Kurganov and G. Petrova (2006). "Finite-Volume-Particle Methods for Models of Transport of Pollutant in Shallow Water." Journal of Scientific Computing **27**(1): 189-199.

Chippada, S., C. N. Dawson, M. L. Martinez and M. F. Wheeler (1998). "A Godunov-type finite volume method for the system of Shallow Water Equations." Computer Methods in Applied Mechanics and Engineering **151**(1): 105-129.

Chu, N. and C. Tai (2005). "Moxi: real-time ink dispersion in absorbent paper." ACM Transactions on Graphics **24**(3): 504-511.

Cobble, M. H. (1973). "Non-Linear Shallow Water Theory." International Journal of Non-Linear Mechanics **8**(6): 513-522.

Cockburn, B. (2003). Discontinuous Galerkin methods. **83:** 731-754.

d'Humieres, D., I. Ginzburg, M. Krafczyk, P. Lallemand and L. S. Luo (2002). "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.

Corel, P. (2004). Wetland Functions and Values.

Dawson, S. P., S. Chen and G. D. Doolen (1993). "Lattice Boltzmann Computations for Reaction-Diffusion Equations." Journal of Chemical Physics **98**(2): 1514-1523.

Dearing, J.A, N. R., A.J Plater, J Wolf, D Prandle, T.J Coulthard (2006). "Modelling approaches for coastal simulation based on cellular automata: the need and potential." Philosophical Transactions of the Royal Society of London Series a-Mathematical Physical and Engineering Sciences 364: 1051-1071.

Deng, J. Q., M. S. Ghidaoui, W. G. Gray and K. Xu (2001). "A Boltzmann-based mesoscopic model for contaminant transport in flow systems." Advances in Water Resources **24**(5): 531-550.

Desplat, J. C., I. Paonabarraga and P. Bladon (2001). "LUDWIG: A parallel Lattice-Boltzmann code for complex fluids." Computer Physics Communications **134**: 273-290.

Duan, Y.-l. and R.-X. Liu (2007). "Lattice Boltzmann simulations of triagular cavity flow and free-surface problems." Journal of Hydrodynamics, Ser. B **19**(2): 127-134.

Einfeldt, B., C. D. Munz, P. L. Roe, B. Sjögreen (1991). On Godunov-type methods near low densities, Academic Press Professional, Inc. **92:** 273-295.

Elder, J. W. (1959). "The dispersion of marked fluid in turbulent shear flow." Journal of Fluid Mechanics Digital Archive **5**: 544-560.

Fagherazzi, S., P. Rasetarinera, M. Y. Hussaini and D. J. Furbish (2004). "Numerical Solution of the Dam-Break Problem with a Discontinuous Galerkin Method." Journal of Hydraulic Engineering **130**(6): 532-539.

Fan, Z., F. Qiu, A. Kaufman and S. Yoakum-Stover (2004). GPU Cluster for High Performance Computing. Proceedings of the 2004 ACM/IEEE conference on Supercomputing, IEEE Computer Society.

Feng, S. D., J. Y. Mao and Q. Zhang (2001). "Lattice Boltzmann equation model in the Coriolis field." Chinese Physics **10**(12): 1103-1105.

Fennema, R. J. and M. H. Chaudhry (1990). "Explicit Methods for 2-D Transient Free Surface Flows." Journal of Hydraulic Engineering **116**(8): 1013-1034.

Fetter, C. W. (1998). Contaminant Hydrology, Prentice Hall.

Frisch, U., B. Hasslacher and Y. Pomeau (1986). "Lattice-Gas Automata for the Navier-Stokes Equation." Physical Review Letters **56**(14): 1505.

Garcia-Navarro, P., F. Alcrudo and J. M. Saviron (1992). "1-D Open-Channel Flow Simulation Using TVD-McCormack Scheme." Journal of Hydraulic Engineering **118**(10): 1359-1372.

García, A., J. A. Revilla, R. Medina, C. Álvarez and J. A. Juanes (2002). "A model for predicting the temporal evolution of dissolved oxygen concentration in shallow estuaries." Hydrobiologia **475-476**(1): 205-211.

Geller, S., M. Krafzyk, J. Tolke, S. Turek and J. Hron (2006). "Benchmark computations based on lattice-Boltzmann, finite element and finite volume methods for laminar flows." Computers and Fluids **35**(8-9): 888-97.

Ghidaoui, M. S., J. Q. Deng, W. G. Gray and K. Xu (2001). "A Boltzmann based model for open channel flows." International Journal for Numerical Methods in Fluids **35**(4): 449-494.

Ginzburg, I. (2005). "Equilibrium-type and link-type lattice Boltzmann models for generic advection and anisotropic-dispersion equation." Advances in Water Resources **28**(11): 1171-1195.

Ginzburg, I. (2006). "Variably saturated flow described with the anisotropic Lattice Boltzmann methods." Computers & Fluids **35**(8-9): 831-848.

Ginzburg, I. (2007). "Lattice Boltzmann modeling with discontinuous collision components: Hydrodynamic and advection-diffusion equations." Journal of Statistical Physics **126**(1): 157-206.

Glorioso, P. D. and A. M. Davies (1995). "The Influence of Eddy Viscosity Formulation, Bottom Topography, and Wind Wave Effects upon the Circulation of a Shallow Bay." Journal of Physical Oceanography **25**(6): 1243-1264.

Gunstensen, A. K., D. H. Rothman, S. Zaleski and G. Zanetti (1991). "Lattice Boltzmann model of immiscible fluids." Physical Review A **43**(8): 4320.

Guo, Z., H. Liu, L.-S. Luo and K. Xu (2008). "A comparative study of the LBE and GKS methods for 2D near incompressible laminar flows." Journal of Computational Physics **227**(10): 4955-4976.

Han, S.-L., P. Zhu and Z.-Q. Lin (2007). "Two-dimensional interpolation-supplemented and Taylor-series expansion-based lattice Boltzmann method and its application." Communications in Nonlinear Science and Numerical Simulation **12**(7): 1162-1171.

Han, Y., Z. Fan, F. Qiu and Y.-C. Kuo (2007). "Visual Simulation of Heat Shimmering and Mirage." IEEE Transactions on Visualization and Computer Graphics **13**(1): 179-189.

Harten, A., P. D. Lax and B. Van Leer (1983). "On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws." SIAM Review **25**(1): 35-61.

He, X., S. Chen and G. D. Doolen (1998). A novel thermal model for the lattice Boltzmann method in incompressible limit, Academic Press Professional, Inc. **146:** 282-300.

He, X., X. Shan and G. D. Doolen (1998). "Discrete Boltzmann equation model for nonideal gases." Physical Review E **57**(1): R13-R16.

Huang, W. and M. Spaulding (1995). "3D Model of Estuarine Circulation and Water Quality Induced by Surface Discharges." Journal of Hydraulic Engineering **121**(4): 300-311.

Inamuro, T., M. Yoshino and F. Ogino (1999). "Lattice Boltzmann simulation of flows in a three-dimensional porous structure." International Journal for Numerical Methods in Fluids **29**(7): 737-748.

Johnson, B. H., K. W. Kim, R. E. Heath, B. B. Hsieh and H. L. Butler (1991). Development and verification of a three-dimensional numerical hydrodynamic, salinity and temperature

model of Chesapeake Bay, U. S. Army Engineer Waterways Experiment Station, Vicksburg, MS.

Kandhai, D., D. J. E. Vidal, A. G. Hoekstra, H. Hoefsloot, P. Iedema and P. M. A. Sloot (1999). "Lattice-Boltzmann and finite element simulations of fluid flow in a SMRX Static Mixer Reactor." International Journal for Numerical Methods in Fluids **31**(6): 1019-1033.

Kang, Q. H., D. X. Zhang and S. Y. Chen (2002). "Unified lattice Boltzmann method for flow in multiscale porous media." Physical Review E **66**(5)

Kang, Q. J., D. X. Zhang, S. Y. Chen and X. Y. He (2002). "Lattice Boltzmann simulation of chemical dissolution in porous media." Physical Review E **65**(3):

Karlin, I. V., A. Ferrante and H. C. ttinger (1999). "Perfect entropy functions of the Lattice Boltzmann method." EPL (Europhysics Letters) **47**(2): 182-188.

Kawahara, M., H. Hirano, K. Tsubota and K. Inagaki (1982). "Selective lumping finite element method for shallow water flow." International Journal for Numerical Methods in Fluids **2**(1): 89-112.

Keming Hu, C. G. M. D. M. C. (2006). "A mesh patching method for finite volume modelling of shallow water flow." International Journal for Numerical Methods in Fluids **50**(12): 1381-1404.

King, I. and W. Norton (1978). Recent application of RMA's finite element models for two-dimensional hydrodynamics and water quality. Finte elements in water resources II. C. Brebbia, W. Gray and G. F. Pinder. London, Pentech Press.

Klar, A., M. Seaid and G. Thommes (2008). "Lattice Boltzmann simulation of depth-averaged models in flow hydraulics." International Journal of Computational Fluid Dynamics **22**(7): 507-522.

Kruger, J. and R. Westermann (2003). "Linear algebra operators for GPU implementation of numerical algorithms." ACM Trans. Graph. **22**(3): 908-916.

Lallemand, P. and L.-S. Luo (2003). "Theory of the lattice Boltzmann method: Acoustic and thermal properties in two and three dimensions." Physical Review E **68**(3): 036706-036730.

Lallemand, P. and L. S. Luo (2000). "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability." Physical Review E **61**(6): 6546-6562.

Li, W., Z. Fan, X. Wei and A. Kaufman (2005). Flow Simulation with Complex Boundaries. GPU Gems II: Programming Techiniques for High-Performance Graphics and General-Purpose Computaiton. M. Pharr, Addison-Wesley**:** 747-764.

Li, Y. and P. Huang (2008). "A coupled lattice Boltzmann model for advection and anisotropic dispersion problem in shallow water." Advances in Water Resources **31**(12): 1719-1730.

Loose, B., Y. Nino and C. Escauriaza (2005). "Finite volume modeling of variable density shallow-water flow equations for a well-mixed estuary: application to the Río Maipo estuary in central Chile." Journal of Hydraulic Research **43**(4): 339-50.

Luettich, R. J., J. J. Westerink and N. W. Scheffner (1991). ADCIRC: An Advanced Three Dimensional Circulation Model for Shelves, Coasts and Estuaries, U.S. Army Corps of Engineers, Washington D.C. 20314-1000.

Lynch, D. R. and W. Gray (1979). "A wave equation model for finite element tidal computations." Computatoinal Fluids **7**: 207-28.

Lynch, D. R. and F. E. Werner (1991). "Three-dimensional hydrodynamics on finite elements. Part II: Non-linear time-stepping model." International Journal for Numerical Methods in Fluids **12**: 27.

Martinez, D. O., S. Chen and W. H. Matthaeus (1994). "Lattice Boltzmann magnetohydrodynamics." Physics of Plasmas **1**(6): 1850-1867.

Martinez, D. O., W. H. Matthaeus, S. Chen and D. C. Montgomery (1994). "Comparison of spectral method and lattice Boltzmann simulations of two-dimensional hydrodynamics." Physics of Fluids **6**(3): 1285-1298.

Massaioli, F. and G. Amati (2002). Achieving high performance in a LBM code using OpenMP. Fourth European Workshop on OpenMP, Roma.

McNamara, G. R. and G. Zanetti (1988). "Use of the Boltzmann Equation to Simulate Lattice-Gas Automata." Physical Review Letters **61**(20): 2332-2335.

Meselhe, E. A., F. Sotiropoulos and F. M. Holly Jr (1997). "Numerical Simulation of Transcritical Flow in Open Channels." Journal of Hydraulic Engineering **123**(9): 774-783.

Navarrina, F., I. Colominas, M. Casteleiro, L. Cueto-Felgueroso, H. Gómez and J. F. A. Soage (2008). "A numerical model for the transport of salinity in estuaries." International Journal for Numerical Methods in Fluids **56**(5): 507-523.

Noble, D. R., J. G. Georgiadis and R. O. Buckius (1996). "Comparison of accuracy and performance for lattice Boltzmann and finite difference simulations of steady viscous flow." International Journal for Numerical Methods in Fluids **23**(1): 1-18.

NVIDIA. (2008). "NVIDIA CUDA Programming Guide 2.0." Retrieved 01 December 2009, from http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf.

P. Ortiz, O. C. Z. J. S. (2006). "Hydrodynamics and transport in estuaries and rivers by the CBS finite element method." International Journal for Numerical Methods in Engineering **66**(10): 1569-1586.

Pohl, T., F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein and T. Zeiser (2004). Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures. Proceedings of the 2004 ACM/IEEE conference on Supercomputing, IEEE Computer Society.

Prendergast, K. H. and K. Xu (1993). Numerical hydrodynamics from gas-kinetic theory, Academic Press Professional, Inc. **109:** 53-66.

Pullin, D. I. (1980). "Direct simulation methods for compressible inviscid ideal-gas flow." Journal of Computational Physics **34**: 231-244.

Qian, Y. H. (1993). Simulating thermohydrodynamics with lattice BGK models, Plenum Press. **8:** 231-242.

Quirk, J. J. (1998). A Contribution to the Great Riemann Solver Debate.

Reitz, R. D. (1981). One-dimensional compressible gas dynamics calculations using the Boltzmann equation. J. Comput. Phys. ; Vol/Issue: 40:2. United States**:** Pages: 108-123.

Roberts, T. W. (1990). The behavior of flux difference splitting schemes near slowly moving shock waves, Academic Press Professional, Inc. **90:** 141-160.

Roe, P. L. and D. S. Balsara (1996). "Notes on the eigensystem of magnetohydrodynamics." SIAM J. Appl. Math. **56**(1): 57-67.

Rothman, D. H. (1988). "Cellular-automaton fluids: A model for flow in porous media." Geophysics **53**(4): 509-518.

Salmon, R. (1999). "The Lattice Boltzmann method as a basis for ocean circulation modeling." Journal of Marine Research **57**: 503-35.

Salmon, R. (1999b). "The Lattice Boltzmann solutions of the three dimensional planetary geostrophic equations." Journal of Marine Research **57**: 847-84.

Sanay, R. and A. Valle-Levinson (2005). "Wind-Induced Circulation in Semienclosed Homogeneous, Rotating Basins." Journal of Physical Oceanography **35**(12): 2520-2531.

Sankaranarayanan, K., I. G. Kevrekidis, S. Sundaresan, J. Lu and G. Tryggvason (2003). "A comparative study of lattice Boltzmann and front-tracking finite-difference methods for bubble simulations." International Journal for Multiphase Flow **29**: 109-116.

Servan-Camas, B. and F. T.-C. Tsai (2010). "Two-Relaxation-Time Lattice Boltzmann Method for Anisotropic Dispersive Henry Problem." Water Resources Research.

Servan-Camas, B. and F. T.-C. Tsai (2008). "Lattice Boltzmann method with two relaxation times for advection-diffusion equation: Third order analysis and stability analysis." Advances in Water Resources **31**(8): 1113-1126.

Servan-Camas, B. and F. T.-C. Tsai (2009). "Non-negativity and stability analyses of lattice Boltzmann method for advection-diffusion equation." Journal of Computational Physics **228**(1): 236-256.

Servan-Camas, B. and F. T.-C. Tsai (2009). "Saltwater intrusion modeling in heterogeneous confined aquifers using two-relaxation-time lattice Boltzmann method." Advances in Water Resources **32**(4): 620-631.

Shan, X. and G. Doolen (1995). Multi-component lattice-Boltzmann model with interparticle interaction.

Shankar, N. J., H. F. Cheong and S. Sankaranarayanan (1997). "Multilevel finite-difference model for three-dimensional hydrodynamic circulation." Ocean Engineering **24**(9): 785-816.

Shinbrot, M. (1970). "Shallow Water Equations." Journal of Engineering Math **4**(4): 293-304.

Simpson, G. and S. Castelltort (2006). "Coupled model of surface water flow, sediment transport and morphological evolution." Computers & Geosciences **32**(10): 1600-1614.

Su, M., K. Xu and M. S. Ghidaoui (1999). Low-speed flow simulation by the gas-kinetic scheme, Academic Press Professional, Inc. **150:** 17-39.

Succi, S. (2001). The Lattice Boltzmann Equation For Fluid Dynamics and Beyond. Oxford, Oxford University Press.

Szymkiewicz, R. (1993). "Oscillation-Free Solution of Shallow Water Equations for Nonstaggered Grid." Journal of Hydraulic Engineering **119**(10): 1118-1137.

Tadeusz, L. (1984). "An interpolation method for an irregular net of nodes." International Journal for Numerical Methods in Engineering **20**(9): 1599-1612.

Takashi, A. (1997). "Derivation of the lattice Boltzmann method by means of the discrete ordinate method for the Boltzmann equation." J. Comput. Phys. **131**(1): 241-246.

Tan, W.-Y. (1992). Shallow Water Hydrodynamics. Amsterdam, Elsevier.

Tao, S. and T. JianHua (2006). "Numerical simulation of pollutant transport acted by wave for a shallow water sea bay." International Journal for Numerical Methods in Fluids **51**(5): 469-487.

Teeter, A. M., B. H. Johnson, C. Berger, G. Stelling, N. W. Scheffner, M. H. Garcia and T. M. Parchure (2001). "Hydrodynamic and sediment transport modeling with emphasis on shallow-water, vegetated areas (lakes, reservoirs, estuaries and lagoons)." Hydrobiologia **444**(1): 1-23.

Thurey, N. and U. Rude (2004). Free surface lattic-Boltzmann fluid simulations with and without level sets. Proceeddings of Workshop on Vision, Modeling, and Visualization. (Stanford, CA), IOS Press, Amsterdam**:** 199-208.

Tolke, J. and M. Krafczyk (2008). "TeraFLOP computing on a desktop PC with GPUs for 3D CFD." International Journal Computational Fluid Dynamics **22**(7): 443-456.

Toro, E. F. (1992). "Riemann Problems and the WAF Method for Solving the Two-Dimensional Shallow Water Equations." Philosophical Transactions of the Royal Society of London Series a-Mathematical Physical and Engineering Sciences **338**(1649): 43-68.

Tseng, M.-H. (2004). "Improved treatment of source terms in TVD scheme for shallow water equations." Advances in Water Resources **27**(6): 617-629.

Tubbs, K. R. and F. T.-C. Tsai (2008). Lattice Boltzmann Modeling Using High Performance Computing for Shallow Water Fluid Flow and Mass Transport. American Geophysical Union 2008 Fall Meeting, San Francisco, CA.

Tubbs, K. R. and F. T.-C. Tsai (2009). A GPU Accelerated Lattice Boltzmann Model for Shallow Water Flow and Mass Transport. Sixth International Conference for Mesoscopic Methods in Engineering and Science (ICMMES), Guangzhou City, Guangdong Province, China.

Tubbs, K. R. and F. T.-C. Tsai (2009). Simulation of Multilayer Shallow Water Fluid Flow Using Lattice Boltzmann Modeling and High Performance Computing, Kansas City, Missouri, ASCE.

Tubbs, K. R. and F. T.-C. Tsai (2009). "Multilayer shallow water flow using lattice Boltzmann method with high performance computing." Advances in Water Resources **32**(12): 1767-1776.

Tubbs, K. R. and F. T.-C. Tsai (2010). "GPU Accelerated Lattice Boltzmann Model for Shallow Water Flow and Mass Transport." International Journal Numerical Methods in Engineering **(under review)**.

Vreugdenhil, C. B. (1994). Numerical Methods for Shallow-Water Flow. Dordrecht, Kluwer Academic Publishers.

Walsh, S. D. C., M. O. Saar, P. Bailey and D. J. Lilja (2009). "Accelerating geoscience and engineering system simulations on graphics hardware." Computers & Geosciences **35**(12): 2353-2364.

Wang, Z. J. (2002). Spectral (finite) volume method for conservation laws on unstructured grids: basic formulation, Academic Press Professional, Inc. **178:** 210-251.

Wei, X., Y. Zhao, Z. Fan, W. Li, F. Qiu, S. Yoakum-stover and A. Kaufman (2004). "Lattice-based flow field modeling." IEEE Transactions on Visualization and Computer Graphics **10**: 719-29.

Wellein, G., T. Zeiser, G. Hager and S. Donath (2006). "On the single processor performance of simple lattice Boltzmann kernels." Computers & Fluids **35**(8-9): 910-19.

Wilke, J., T. Pohl, M. Kowarschik and U. Rude (2003). Cache Performance Optimizations for Parallel Lattice Boltzmann Codes. Euro-Par 2003 Parallel Processing**:** 441-450.

Wolf-Gladrow, D. A. (2000). Lattice Gas Cellular Automata and Lattice Boltzmann Models: an introduction. Berlin, Springer.

Wu, W. (2004). "Depth-Averaged Two-Dimensional Numerical Modeling of Unsteady Flow and Nonuniform Sediment Transport in Open Channels." Journal of Hydraulic Engineering **130**(10): 1013-1024.

Xu, K. (1997). BGK-based scheme for multicomponent flow calculations, Academic Press Professional, Inc. **134:** 122-133.

Xu, K. (1999). A Gas-Kinetic Scheme for the Euler Equations with Heat Transfer, Society for Industrial and Applied Mathematics. **20:** 1317-1335.

Xu, K., C. Kim, L. Martinelli and A. Jameson (1996). BGK-Based Schemes for the Simulation of Compressible Flow, Taylor & Francis. **7:** 213 - 235.

Xu, K., L. Martinelli and A. Jameson (1995). "Gas-kinetic finite volume methods, flux-vector splitting, and artificial diffusion." J. Comput. Phys. **120**(1): 48-65.

Yoon, T. H. and S. K. Kang (2004). "Finite Volume Model for Two-Dimensional Shallow Water Flows on Unstructured Grids." Journal of Hydraulic Engineering **130**(7): 678-688.

Zhang, X., A. G. Bengough, L. K. Deeks, J. W. Crawford and I. M. Young (2002). "A novel three-dimensional lattice Boltzmann model for solute transport in variably saturated porous media." Water Resour. Res. **38**.

Zhao, Y. (2008). "Lattice Boltzmann based PDE solver on the GPU." The Visual Computer **24**(5): 323-333.

Zhao, Y., F. Qiu, Z. Fan and A. Kaufman (2007). Flow simulation with locally-refined LBM. Proceedings of the 2007 symposium on Interactive 3D graphics and games. Seattle, Washington, ACM.

Zhong, L., S. Feng, D. Lou and S. Gao (2006). "Wind-driven, double-gyre, ocean circulation in a reduced-gravity, 2.5-layer, lattice boltzmann model." Advances in Atmospheric Sciences **23**(4): 561-78.

Zhou, J. G. (2002). "A lattice Boltzmann model for the shallow water equations." Computer Methods in Applied Mechanics and Engineering **191**(32): 3527-3539.

Zhou, J. G. (2002). "A Lattice Boltzmann Model for the Shallow Water Equations with Turbulence Modeling." International Journal of Modern Physics C: Computational Physics & Physical Computation **13**(8): 1135.

Zhou, J. G. (2004). Lattice Boltzmann Methods for Shallow Water Flows, Springer.

Zhou, J. G. (2007). "Lattice Boltzmann Simulations of Discontinuous Flows." International Journal of Modern Physics C: Computational Physics & Physical Computation **18**(1): 1-14.

Zhou, J. G., D. M. Causon, C. G. Mingham and D. M. Ingram (2004). "Numerical Prediction of Dam-Break Flows in General Geometries with Complex Bed Topography." Journal of Hydraulic Engineering **130**(4): 332-340.

Zienkiewicz, O. C. and R. Codina (1995). "A general algorithm for compressible and incompressible flow Part I. the split, characteristic-based scheme." International Journal for Numerical Methods in Fluids **20**: 869-885.

Zienkiewicz, O. C. and P. Ortiz (1995). "A split-characteristic based finite element model for the shallow water equations." International Journal for Numerical Methods in Fluids **20**(8-9): 1061-1080.

Zou, Q. and X. He (1997). "On pressure and velocity boundary conditions for the lattice Boltzmann BGK model." Physics of Fluids **9**(6): 1591-1598.

VITA

Kevin Tubbs was born in Baton Rouge, Louisiana, in April, 1979, to John and Veronica Tubbs. He pursued his high school education at Southern University Laboratory School. He received his bachelor's of science degree in physics for Southern University, Baton Rouge, in May 2001. In May 2004, He received his master of science degree in physics from Louisiana State University. He then entered the doctoral program in the Donald W. Clayton Engineering Science program. Under the advice of Dr. Frank T.-C. Tsai, he conducted research in the field of hydrodynamics and high performance computing, focusing on numerical modeling using lattice Boltzmann techniques.