Efficient Tsunami Simulation at Local and Global Scales

Xinsheng Qin

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Michael R. Motley, Chair

Randall J. LeVeque, Chair

Frank I. Gonzalez

Program Authorized to Offer Degree:
Civil and Environmental Engineering

University of Washington

Abstract

Efficient Tsunami Simulation at Local and Global Scales

Xinsheng Qin

Co-Chairs of the Supervisory Committee:
Professor Michael R. Motley
Civil and Environmental Engineering

Professor Randall J. LeVeque
Applied Mathematics

Tsunami hazard evaluation and mitigation is of great importance to coastal communities around the world, especially after the frequent occurrence of large tsunamis in the past two decades. Many physical phenomena need to be modeled during a tsunami event, e.g. tsunami wave generation and propagation, coastal inundation, and forces on structures. Most of them are nonlinear and involve a wide range of length scales, and thus are challenging to model. In this dissertation, the ability of three-dimensional (3D) and two-dimensional (2D) models to capture tsunami forces on structures and flow through a constructed environment is first analyzed. Then the development of a GPU-accelerated hyperbolic partial differential equation (PDE) solver with adaptive mesh refinement (AMR), with application to solving several PDEs that govern different physical processes arising in tsunamis, is presented and discussed.

Tsunami inundation is the final and most destructive phase of tsunami evolution that comes after tsunami wave propagation in the ocean. The numerical modeling of this phase that incorporates the constructed environment of coastal communities is challenging for both 2D and 3D models. Inundation and flooding in this region can be too complex for 2D models to capture properly, while for 3D models a very fine mesh is required to properly capture the physics, dramatically increasing the computational cost and rendering impractical modeling of some problems. To evaluate the

capability of the current tsunami inundation models, comparisons are made between GeoClaw, a depth-integrated 2D model based on the nonlinear shallow water equations (NSWE), and the interFoam solver in OpenFOAM, a 3D model based on Reynolds Averaged Navier-Stokes (RANS) equations for tsunami inundation modeling. The two models are first validated against existing experimental data of a bore impinging onto a single square column. Then they are used to simulate tsunami inundation in a physical wave tank model of Seaside, Oregon. The resulting flow parameters from the models are compared and discussed, and these results are used to extrapolate tsunami-induced force predictions and give guidance for the use of numerical models in other similar situations.

Numerical modeling of tsunami processes is computationally expensive. Being able to do this faster means we can simulate a problem with higher resolution to potentially get more accurate result, simulate the same problem faster to send out tsunami warning earlier, or perform more tsunami simulations within a given time budget when doing probabilistic hazard assessment or studying the uncertainties of the process. Using Adaptive Mesh Refinement (AMR) as implemented in GeoClaw speeds up the process by greatly reducing computational demands, while accelerating the code using the Graphics Processing Unit (GPU) could do so through faster hardware but has not previously been implemented in GeoClaw. The second part of this dissertation presents an efficient CUDA implementation of the GeoClaw code. The code can model transoceanic tsunami simulation by using AMR and solving the shallow water equations in spherical coordinates. Numerical experiments of the 2011 Japan tsunami and a local tsunami triggered by a hypothetical Mw 7.3 earthquake on the Seattle Fault illustrate the correctness and efficiency of the code. The GPU implementation, when running on a single GPU, is observed to be 3.6 to 6.4 times faster than the original model running in parallel on a 16-core CPU. Three metrics are proposed to evaluate performance of the model, which shows efficient usage of hardware resources.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like express my gratitude to all of the people who have made this thesis possible. First I would like to thank my advisors, Michael Motley, Randy LeVeque and Frank Gonzalez, for their continuous guidance in the past five years. They all have been extremely patient with not only the difficulties I ran into during my research, but also the difficulties I had as an international student and as a non-native English speaker. Michael has been very flexible with what I would like to explore and has provided many research ideas. Randy has always been incredibly helpful, encouraging and supportive. Without Randy, many of the research ideas I would like to study would not be possible.

I also would like to thank many professors and researchers I have met throughout the years who have influenced my mind, my character and my career decisions. Thank you to professors Antonino Ferrante and professors Duane Storti who not only served as my committee members but also taught me in classes that form the fundamental of my researches. I would like to thank Dr. Weiqun Zhang and Dr. Ann S. Almgren who were my mentors during an internship at the Berkeley Lab. This experience and discussion with them have inspired many research ideas and have influenced many of the works in the dissertation. I also would like to thank Dr. Yong Wei and Dr. Diego Arcas from National Oceanic and Atmospheric Administration (NOAA) for discussion of research ideas and their advices on my career. Professor Loyce Adams has always made me feel at home with her excellent cooking skills. I also would like to thank you for all the delicious food that I would not have had the chance to taste without you.

Finally, I would like to thank my family and all of my peers. Thank you to my father and mother for constant support and love. I am extremely grateful for your patience and for all you have done in bringing me to be the person I am today. Thank you to my colleagues and friends

at the University of Washington, Molly Grear, Han-Gyu Kim, Donsub Rim, Andrew O. Winter, Tianye (Andrew) Yang, for your help and companion throughout the graduate program.

DEDICATION

to my father and mother

Chapter 1

INTRODUCTION

In the last two decades, tsunami hazards across the Pacific Ocean have resulted in hundreds of thousands of fatalities and significant infrastructure damages. The 2004 Indonesia tsunami impacted more than 10 countries and killed about 230,000 people [Imamura et al., 2007]. The 2011 Tohoku earthquake tsunami moved inland more than 5 km in some locations and had a maximum inundation height of 19.5 m, ultimately causing 15,641 fatalities and 5,007 missing people [Mori et al., 2011]. More recently, the Sulawesi tsunami and Sunda Strait tsunami hit Indonesia at the end of 2018 within two months. The Sulawesi tsunami generated waves that stood as tall as 18 feet, causing severe casualties and damages (about 2,000 dead, 4,000 injured and 1,000 missing) [Wei-Haas, 2018] while the Sunda Strait tsunami killed 437 and injured 14,059 [Plus, 2018].

In the western United States, the United States Geological Survey has identified major tsunami hazards for the states of Alaska, California, Hawaii, Oregon, and Washington. In Washington State, all marine shorelines are vulnerable to tsunamis. Geologic evidence has been found on the Pacific coast and in the Strait of Juan de Fuca and Puget Sound, indicating future tsunamis are inevitable. The Cascadia Subduction Zone (CSZ) fault that runs from northern California to Vancouver Island is known to have experienced roughly 20 Magnitude 9 earthquakes and numerous Magnitude 8–8.5 earthquakes in the past 10,000 years, which can trigger devastating tsunamis [Atwater et al., 2016, Goldfinger et al., 2012, Leonard et al., 2010]. It is reported that the region could have an earthquake of magnitude 8 or greater in the next 50 years with a probability of 10-14% [Petersen et al., 2002].

Development of efficient, accurate and robust tsunami models that are able to simulate the entire tsunami process plays a key role in tsunami hazards mitigation. These models can be used for

1. a tsunami early warning system

2. design of buildings and other structures in a tsunami-prone area

3. Probabilistic Tsunami Hazard Assessment (PTHA).

Tsunami early warnings for the Pacific Northwest are issued by the National Tsunami Warning Center in Palmer, AK, operated by NOAA. The warning system currently relies on a combination of seismic information used to estimate the location and magnitude of the earthquake, and sea surface elevation data measured by DART (Deep-ocean Assessment and Reporting of Tsunami) buoys at a sparse set of locations in the deep ocean [for Tsunami Research, Titov et al., 2005]. These data are used in real time to perform source inversion to estimate the tsunami source and to compute sea floor deformation, which can be taken as input to a tsunami model that simulates the tsunami propagation and inundation. Once the simulation is done, the simulation results provide arrival time and severity of potential tsunami hazards at locations of interest. Such an early warning system works well for providing real-time tsunami warnings in the Pacific Northwest for far-field tsunamis, such as those originating in Japan, Alaska, or Chile for example, which often take hours to hit the West coast of the United States. As a key component of such an early warning system, a tsunami model should run as fast as possible, which can give earlier warning and more response time to emergency managers, citizens, and first responders in coastal regions.

Depending on the distance from a tsunami source (e.g., an undersea earthquake that triggers a tsunami) to the coast, an early warning system might not be able to send out tsunami warnings early enough, or waves can arrive before people can evacuate away from the inundation zone even if they are warned immediately. For these reasons, many coastal communities are building vertical evacuation structures that provide a safe evacuation place within inundation zones. These structures must be designed to withstand the significant fluid forces that a tsunami will impose on them. Other critical structures such as hospitals or fire stations also must be designed to survive these events. Accurate predictions of the inundation depth and flow speed around these buildings, and of the corresponding fluid forces on them, are necessary for the design and construction of

these buildings and structures. The latest version of ASCE 7-16 in the United States has a chapter for tsunami loads and effect for coastal structures. The provision requires site-specific inundation modeling and analysis be performed for all vertical evacuation structures. One of such examples is the design of the first vertical evacuation structure in the United States [Ash, 2015], the site-specific inundation analysis of which was conducted by González et al. [2013].

Geist and Parsons [2006] first adapted the 1968 Cornell Probabilistic Seismic Hazard Analysis (PSHA) methodology [Cornell, 1968] to PTHA in 2006. The study takes into account the uncertainties in tsunami hazards assessment and produces hazard curves and/or hazard maps for regions of interest. The hazard curve at a point shows a quantity of interest (e.g., the maximum depth of flooding) on the horizontal axis and the probability of exceeding that value on the vertical axis. The hazard map shows a quantity of interest at each point in regions of interest for a fixed probability. Another variance of hazard map shows the probability of a quantity of interest exceeding a fixed threshold at each point in regions of interest. To construct such hazard curves and hazard maps, suites of possible earthquake scenarios for a given region developed by geological and seismological experts are used as inputs to tsunami models. The tsunami models then produce a set of quantities of interest at the study region for different earthquake scenarios, which are used to construct the hazard curves and hazard maps. The efficiency of tsunami models is critical in PTHA because a faster tsunami model allows more simulations in a fixed time budget and thus can construct higher-fidelity hazard curves and hazard maps.

This dissertation contributes to multiple aspects of the study of tsunami models described above and is divided into two parts. Part I of the dissertation presents and compares a three-dimensional (3D) and a two-dimensional (2D) tsunami inundation models, with comparison to experimental measurements from a wave tank experiment of a tsunami impacting a 1:50 scale model of an idealized community, representative of Seaside, Oregon [Park et al., 2013]. The explicitly represented constructed environment adds complexity and challenge to the problem. In particular, the 3D model is built with OpenFOAM, an open-source Computational Fluid Dynamics (CFD) package that can solve Reynolds Averaged Navier-Stokes (RANS) equation, while the 2D model is based on GeoClaw, an open-source code that solves the nonlinear shallow water equations. Flow param-

eters – water level, velocity, and momentum flux – at several locations among the buildings are validated against experimental data. The agreement in these quantities between numerical results and experiment measurements are good, except for peak value of velocity and momentum flux. This part is primarily based on three published papers, Qin et al. [2018c,b, 2017].

Chapter 3 describes the setup and results from the 3D model. The entire wave basin can not be modeled as a whole by the 3D model due to limitation in computational resources. Instead, it is modeled by using subsections with proper width without loss of accuracy in areas of interest. The discrepancies in the peak value of velocity are analyzed in detail to reveal the potential problematic assumption in the approach to measuring peak velocity in the experiment. It is found that the optical method used in the experiment to predict the peak velocities might underestimate the peak velocity because it assumes the velocity of the leading edge of the bore corresponds to the maximum flow speed, which is shown to be problematic by the numerical results. This underestimation in peak velocity can cause large underestimation in forces as the later is usually in proportion to the square of velocity.

Chapter 3 further discusses how we can predict fluid forces on coastal structures from numerical results. The direct approach can do so by integrating pressure and shear stress on their walls while the indirect approach is based on the definition of the drag coefficient and uses flow parameters from the numerical results to extrapolate fluid forces on coastal structures. The amplitude and direction of local forces on structures and influence of constructed environments on these forces are also shown.

In chapter 4, a 2D model based on GeoClaw for the same wave tank experiment is developed. Flow parameters at the same locations are measured from the 2D numerical simulation for comparison with those from the previous 3D model. The predicted flow parameters from the 2D model agree well with experimental measurements in the time region after the initial impact of the flow on structures at most gauges. Near the initial-impact time region, the 2D GeoClaw model has more difficulty in predicting the flow parameters due to transient characteristic of the flow. Although the 3D OpenFOAM model can do better in capturing these turbulent flows, it does so at an expense of much more computational resources. The 2D GeoClaw model requires much less computational

resources, can run much faster and can model the entire wave basin as a whole at the same time. The same approach based on the drag coefficient is used to extrapolate prediction of forces from the 2D numerical results since the 2D model can not produce the pressure field that is required by the direct approach. The limitation of this approach is discussed and its outcome for the 2D model is compared against the 3D model results. Trade-offs between the two models due to their different levels of accuracy and required computational resources are thoroughly discussed in this chapter.

The remainder of the dissertation (part II) presents a GPU-accelerated version of the open source GeoClaw code used in the first part of the dissertation and is primarily based on two published papers, Qin et al. [2018a, 2019]. The source code developed in the study is publicly available on Github at https://github.com/xinshengqin/geoclaw/tree/geo_gpu_paper and on Zenodo at https://doi.org/10.5281/zenodo.2727368. It is also being freely incorporated into the Clawpack package [Clawpack Development Team] and is described on www.clawpack.org/gpu.html.

The CPU version of GeoClaw is parallelized with OpenMP on multicore shared memory machines, and this part of the dissertation describes how this has been extended to use CUDA-based GPU acceleration [Nickolls et al., 2008] on such a machine. Due to the use of adaptive mesh refinement (AMR) in GeoClaw, many realistic modeling problems can be solved on such hardware without the need for distributed memory parallelization. The GPU acceleration allows an additional increase in speed that may be particularly useful for PTHA applications and early warning systems.

Chapter 5 gives motivation and related work for the study described in part II. In particular, multiple variants of the AMR algorithm and relevant literatures are briefly summarized. The previous works that are relevant to either AMR or GPU-accelerated solvers on regular uniform grids are discussed. They highlight the challenges and significance in combining AMR and GPU acceleration, which is the primary goal of this part.

Chapter 6 summarizes the relevant numerical methods and the AMR algorithm implemented in GeoClaw to the extent needed to explain the GPU implementation, which is described in chapter 7.

In chapter 7, details of the philosophy behind the GPU code design and key implementations are presented. In particular, a hybrid CPU/GPU approach is used to utilize the computational resource from both the CPU and the GPU. A custom memory pool is used to reduce the overhead of frequent memory operations. The chapter then introduces the CUDA programming model and describes the CUDA kernel developed for advancing solutions on the computational grids.

Chapter 8 describes the benchmark experiments conducted to evaluate the performance of the GPU implementation. First, the hardware used for the experiments is introduced and some metrics are proposed to facilitate performance evaluation. Two realistic tsunami modeling problems from recent validation studies are then simulated by the GPU-accelerated GeoClaw. The first problem uses AMR to model waves propagating across the ocean from the 2011 Japan Tohoku tsunami, along with fine grid modeling around the tide gauge at Crescent City, CA in the United States. The second problem is a local tsunami triggered by the Seattle Fault, with AMR used to focus on inundation in a coastal region very close to the fault, so the finest grid levels are activated immediately. With the GPU, the entire tsunami model runs 3.6 to 6.4 times faster than an original CPU-based GeoClaw tsunami model for the two benchmark problems. In particular, the Japan 2011 Tohoku tsunami can be fully simulated for 13 hours in under 3.5 minutes wall-clock time, using a single Nvidia TITAN X GPU. This work can also benefit the numerical modeling of other hazards such as storm surge (e.g. Mandli and Dawson [2014]) and dam failures (e.g. George [2011]) which can also be modeled with GeoClaw.

Part I

THREE-DIMENSIONAL AND TWO-DIMENSIONAL TSUNAMI INUNDATION MODELS

Chapter 2

PREVIOUS WORK ON TSUNAMI MODELS

For many years, researchers have been working on different numerical models with the focus on different phases that are of very different scales (from thousands of kilometers to tens of meters), including tsunami generation from the source [e.g., Nosov, 2014, Okada, 1985], long-distance (transoceanic) propagation [e.g., George, 2006, Choi et al., 2003, Titov and Gonzalez, 1997], local inundation in coastal regions [e.g., Park et al., 2013, Qin et al., 2017, 2018b] and the interaction of water waves with coastal structures [e.g., Motley et al., 2015, Qin et al., 2018c, Winter et al., 2017]. Some tsunami models use separate sub-models for different phases of tsunamis, while some integrate the modeling of multiple phases into a single simulation [e.g., Zhang and Baptista, 2008, Macías et al., 2016], facing the computational challenges induced by very different scales in the problem.

2.1 Tsunami Propagation and Runup on Bare Earth

Earlier study on tsunami modeling has largely focused on tsunami wave propagation, and on maximum runup heights or the free surface elevations on simple coastal topographies or bare earth without modeling the complex flow in constructed environments. Among these, there are several prevailing benchmark studies. Synolakis [1987] proposed an approximate theory for runup of non-breaking solitary waves on plane beaches and provided physical experimental data to validate his theory. Kânoğlu and Synolakis [1998] studied the evolution of long-wave runup on compound sloped beaches. The analytical methods they developed were able to predict tsunami runup and rundown processes and the maximum runup heights, which agreed well with both numerical and experimental results. Briggs et al. [1995] conducted a set of laboratory experiments to study three-dimensional (3D) tsunami wave runup on conical island and provided time histories of free-surface

elevation and runup histories. Titov and Synolakis [1998] presented a numerical model to solve the 2 + 1 nonlinear shallow water equations without friction factors or artificial viscosity, and it was proved to be able to predict wave runup and overland flow. Carrier et al. [2003] presented an analytical-numerical hybrid method to compute tsunami runup and rundown on a simple beach with uniform slope based on fully nonlinear shallow-water wave theory and also discussed the occurrence of maximum velocity and momentum flux under different initial conditions.

In the past decades, many other models have been developed for these tsunami phases. These tsunami models can be categorized into two-dimensional (2D) models and three-dimensional models. Due to large differences in scale for different phases of a tsunami, many tsunami models solve 2D depth-integrated equations, e.g., the nonlinear shallow water equations (NSWE) or Boussinesq-type wave equations, often with computational grids that vary in spatial resolution from the order of several kilometers in the ocean to the order of several meters near the coast. Some 2D models can be used in the near-shore and inundation zone, since they can handle non-linearity that arises in very shallow water and can be adapted to deal robustly with wetting and drying. However, it is not clear whether these models are adequate to properly resolve 3D turbulent flow in a smaller scale, particularly in the scale necessary to determine tsunami impact and corresponding tsunami-induced forces on individual structures. The 3D models are often based on the 3D Navier-Stokes equations with a turbulence model. However, it is extremely expensive to solve the Navier-Stokes equations in this scale compared to 2D models, and is only practical for detailed simulations over small spatial regions.

Among these tsunami models, the 2D models are the most widely used for their simplicity and computational efficiency. Popinet [2012] simulated the 2011 Tohoku tsunami by solving the 2D NSWE with dynamically-adapted spatial resolution that varied from 250 m in flooded areas near-shore up to 250 km offshore. The model accurately predicted long-distance wave and coarse-scale flooding; the initial surface elevation was determined from a source model based on seismic inversion (as opposed to inversion of DART buoys and tidal gauge time series). This also showed that an accurate and consistent model of tsunami wave propagation can sometimes be constructed using only seismic wave inversion. Wei et al. [2013] used the Method of Splitting Tsunamis (MOST)

model to model the same tsunami event. The MOST model solves the shallow water equations in spherical coordinates with numerical dispersion. Their results demonstrated that it may be possible to forecast near-field tsunami inundation in real time. The GeoClaw software [Berger et al., 2011, LeVeque et al., 2011] has been developed and used for the simulation of many tsunamis. It features the adaptive mesh refinement algorithm to save computational cost and can solve the shallow water equations on either regular Cartesian grids for local tsunamis or on spherical coordinates for global-scale tsunamis [Arcos and LeVeque, 2015, MacInnes et al., 2013, Galanti et al., 2011, Adams and LeVeque, 2017]. Hu et al. [2000] presented an NSWE model that can simulate storm waves propagating in the coastal surf zone and overtopping a sea wall. They found that waves overtopping a vertical wall may be approximately modeled by representing the wall as a steep slope, and that the overtopping rate was sensitive to the bottom friction and the minimum friction depth. The 2D NSWE model of wave run-up and overtopping by Hubbard and Dodd [2002] featured an adaptive mesh refinement algorithm. Their model can accurately reproduce one-dimensional and 2D wave transformation, run-up and overtopping in physical experiments. Their modeling of seawall overtopping by off-normal incident waves showed that there could be more flooding in such a situation than at normal incidence. Lynett [2007] simulated long wave runup obstructed by an obstacle and concluded that the obstacle could help reduce runup and maximum overland velocity if the wave is highly nonlinear (with a ratio of wave height to shelf water depth ≥ 0.5). The sensitivity study also showed that in cases of breaking waves, the Boussinesq model was more accurate than the nonlinear shallow water equations in terms of wave runup (maximum differences up to 10%). For nonbreaking long waves, differences between the two were negligible. Shi et al. [2012] developed a high-order adaptive time-stepping TVD solver for a fully nonlinear Boussinesq model and validated it against a series of laboratory experiments for wave shoaling and breaking and a suite of benchmark tests for wave runup. The results showed that the model was able to accurately model wave shoaling, breaking, and wave-induced near-shore circulation. With a Boussinesq model, Lynett et al. [2010] simulated overtopping of levees of the Mississippi River-Gulf Outlet (MRGO) during Hurricane Katrina at four several characteristic transects along the 20 km-long stretch of the levees. The predicted overtopping rates agreed well with the observed data.

As computing power increases, it becomes possible to model the tsunami runup process by solving the 3D Navier-Stokes equations with a proper turbulence closure. Choi et al. [2007] solved the 3D Reynolds Averaged Navier-Stokes (RANS) equations to simulate wave runup on an conical island and compared different turbulence closure models including $k-\epsilon$, RNG (Re-Normalisation Group methods, Yakhot et al. [1992]) $k-\epsilon$ and LES (Large Eddy Simulation). Their results showed that LES and RNG $k-\epsilon$ are similar and more accurate than $k-\epsilon$. Williams and Fuhrman [2016] solved incompressible RANS equations with a transitional variant of the standard two-equation $k-\omega$ turbulence closure to study boundary layer flow induced by tsunami-scale waves. Their results indicated that the boundary layer generated by a tsunami is both current-like due to the long duration and wave-like due to its unsteadiness. The study also indicated that an existing expression for the maximum bed shear stress under the wind wave scale can be reasonably extrapolated to full tsunami scale. Mayer and Madsen [2001] investigated wave breaking in the surf zone by solving the RANS equations with a $k-\omega$ turbulence model. They found that the volume-of-fluid method could be used successfully to simulate wave breaking and that although some instabilities occurred in applying the RANS equations, they could be eliminated by an ad-hoc modification of the turbulence model.

2.2 Tsunami Inundation in Constructed Environments

While the studies above provide valuable information, to understand tsunami risk for coastal communities, the study of tsunami wave runup and rundown on simple topography or bare earth is insufficient. It is shown by some field surveys that near-shore structures like seawalls and buildings have a significant influence on hydrodynamics of the tsunami event and can reduce tsunami damage [Darlymple and Kriebel, 2005, Tomita et al., 2006, Yeh et al., 2013]. Beyond the field surveys, some experimental and numerical studies have been conducted to model such interactions between tsunami inundation and onshore structures. Lynett [2007] numerically studied how shallow shelf obstacles affect the nonlinear long wave runup. Their 2D model was applicable only to near shore regions with simple topography like those characterized by long and continuous reefs or breakwaters. But 3D effects due to alongshore breaks, as observed in tsunami field studies [e.g.

Liu et al., 2005], can lead to significantly increased runup and flow velocities. Tomita and Honda [2007] also built a numerical model with such macro-roughness onshore to investigate inundation, which agreed well with observations in Galle city, Sri Lanka from the 2004 Indian Ocean tsunami. More recently, researchers conducted a series of studies on tsunami inundation in constructed environments, both experimentally and numerically: a single tsunami long wave propagated onto a compound sloped beach, upon which a $1/50$ scale model of a coastal town with nearly 100 houses and buildings was constructed. Surface elevation, velocity and momentum flux around those structures and the effects of constructed environments on flow through the community were measured and discussed [Cox et al., 2008, Rueben et al., 2011, Shin et al., 2012, Park et al., 2013].

From the numerical modeling perspective, inclusion of constructed environments makes modeling the process more challenging for both 2D models and 3D models. The 2D models are good for inundation on bare earth but inclusion of the explicitly represented constructed environments increases the complexity of the topography and the flows become more 3D with large variation in vertical direction and with transient and turbulent flow impacting the structure. Ozer Sozdinler et al. [2015] used the numerical code NAMI DANCE to investigate tsunami inundation hydrodynamic parameters in inundation zones with idealized structures – three rows of 20 blocks representing three-story concrete buildings. The code solved the NSWE using a finite-difference technique in a staggered leapfrog scheme. The effect of wave period, wave shape, protection structures, building layout and Manning's friction coefficient were discussed. Some major conclusions included that the coastal protection structures like seawalls and breakwaters had very limited effect if the waves are able to overtop them and that it was preferable to use different Manning's coefficients for the sea, land and buildings if more accurate values of hydrodynamic parameters are needed, but at the expense of more computational time. Similar conclusions on the Manning's coefficient were presented by Park et al. [2013]. They simulated tsunami inundation in part of Seaside, Oregon and compared flow parameters with their physical experiment. The comparison showed that the flow parameters were sensitive to the friction coefficient, especially for the momentum flux, which is proportional to tsunami loads on structures. For instance, decreasing the friction coefficient by a factor of 10 increased the predicted momentum flux by 208%. Muhari et al. [2011] compared three

different tsunami inundation models for evaluating tsunami impact on coastal communities: 1) a Constant Roughness Model (CRM) which uses a constant friction coefficient, does not include constructed environments and assumes that all buildings are not able to withstand the tsunami; 2) a Topographic Model (TM) which includes constructed environments by incorporating building shape and height information into the topography; 3) an Equivalent Roughness Model (ERM) which represents the building by using a different equivalent friction coefficient at the site of a building on the original topography (with only terrain information but not building height). Both the TM model and the ERM model gave more reliable prediction than the CRM model did, which confirmed the importance of taking constructed environments into consideration.

For 3D models that solve the Navier-Stokes equations, fine mesh needs to be generated around each individual structure, which dramatically increases number of cells in the computational domain and thus computational cost. Shin et al. [2012] applied a two-phase 3D LES (Large Eddy Simulation) model to simulate inland tsunami inundation in a coastal city with hundreds of buildings and compared the prediction with experimental measurements. However, a fairly coarse mesh was used on land and each building had only 3 to 5 mesh cells along its edge in the along-shore or cross-shore direction, so that the resulting agreement in flooding depth can only be considered qualitative.

Being able to predict potential damage to coastal communities is also necessary, and structures that are expected to play indispensable roles in vertical evacuation and recovery efforts need to be designed with these considerations. Ramsden [1993] conducted experimental studies on the slamming force (also called surging force, caused by fluid initially impacting an object) on a wall due to bores, dry-bed surges and steep solitary waves. He found that such forces were determined by the slope of front profile of bore. For a steep front, the drag coefficient to evaluate force could increase from approximately 2.0, which is the suggested value for a square shaped column, to around 3.0. Yeh [2006, 2007] provided a detailed review of tsunami forces on coastal structures and evaluated tsunami forces on objects in a wave runup zone by multiplying momentum flux with a drag coefficient. They also provided a maximum tsunami force distribution in a runup zone by forming an envelope of the momentum flux obtained from existing analytical and numerical solu-

tions. Such a formula for computing hydrodynamic forces is for steady flows; runup and rundown flows in the study was assumed to be quasi-steady. A limitation of their study was that it assumed a one-dimensional flow field with no lateral variation, a uniformly sloping beach and inviscid fluid. Xiao and Huang [2008] simulated the impact of a solitary wave on a beachfront house located at different elevations on a plane beach and found that with every increase of elevation of the house by $1/4$ of maximum vertical runup height of the tsunami wave, the maximum force and moment on the house were reduced to approximately half of the value at the previous elevation. Lindt et al. [2009] measured tsunami forces on a $1/6$ scale model of wood residential building in a laboratory test and studied how some common design features affect the measured forces. They found that tsunami wave loading may produce an uplift load on raised structures (water was allowed to go underneath them). These uplift forces could be reduced due to the weight of water if windows are open or broken such that water can enter the building.

Chapter 3

THE THREE-DIMENSIONAL SEASIDE MODEL

Recent research on tsunami inundation of coastal communities has largely focused on numerical modeling of tsunami runup on one-dimensional slopes or two-dimensional (2D) topographies with more complex shorelines. The coastal bathymetry that are generally applied are based on bare earth topography. Few studies have simulated overland flow around existing macro-roughness features such as near-shore buildings and bridges (also referred to as constructed environments or built infrastructure). This is often due to both lack of available data and the fact that it can be computationally demanding to incorporate these additional features. A consequence of omitting the built infrastructure from models is that, to predict tsunami forces and corresponding structural demand, extrapolations from overland flow data is necessary, and for densely populated communities these flow predictions may not be appropriate. In this chapter, a high-resolution three-dimensional (3D) computational fluid dynamics (CFD) approach is used to model tsunami inundation in a 1:50 model-scale, idealized community, representative of the town of Seaside, OR, USA, for which existing experimental data is available [Park et al., 2013]. Several buildings were selected, and flow characteristics—water level, velocity, and momentum flux—in the vicinity of these buildings were validated against experimental data. Several computational domains were analyzed and it is shown that local flow characteristics can be captured using sub-domains of the community, reducing computational demand. Force predictions using direct integration of predicted pressure were compared with bare earth momentum flux calculations, and the importance of incorporating constructed environments in force prediction models is discussed.

3.1 Methodology

In this chapter, the open-source CFD package OpenFOAM (version 2.3.1) [The OpenFOAM Foundation, 2014b], was used for all simulations. Specifically, the interFoam solver was applied to model two immiscible, incompressible fluids (water and air in this case) with a free interface between them, using a volume-of-fluid (VOF) approach. In this method, a scalar field α_{water} is defined to represent volume fraction of water in each cell. A cell with $\alpha_{water} = 1.0$ indicates that it is completely filled with water (ρ = 1000 kg/m^3, $\nu = 1.0 \times 10^{-6}$ m^2/s) while a cell with $\alpha_{water} = 0.0$ indicates it is completely filled with air (ρ = 1.22 kg/m^3, $\nu = 1.48 \times 10^{-5}$ m^2/s), where ρ is the mass density of the fluid and ν is the kinematic viscosity. A cell with α_{water} between 0 and 1 marks a interface cell. A special transport equation is used to advance the α_{water} field. To model the turbulence, Reynolds-averaged Navier Stokes (RANS) equations with Menter's k-ω-SST model [Menter and Esch, 2001] were solved by the interFoam solver.

The k-ω-SST model was chosen for this work because the k-ϵ model is suitable for fully turbulent flow and non-separated flows and has the shortcoming of numerical stiffness in the viscous sublayer, which can result in stability issues [Menter, 1993]. The k-ω-SST generally behaves better in modeling partially separated flows, which is the case in the current Seaside model (flow becomes separated after passing around the constructed environments), and has significant advantages in numerical stability. In the Seaside model described later in this chapter, the k-ϵ model was also applied but stability issues occurred.

The necessity of using a turbulence model was verified by modeling a similar but simpler case where a bore initiated by a dam-break propagated along a channel and interacted with a free-standing coastal structure downstream. Experimental results are available from Arnason [2005], who performed such an experiment at the Charles W. Harris Hydraulics Laboratory at the University of Washington (UW), Seattle. In the experiment, a square column was placed in a 16.6 m long, 0.6m wide and 0.45 m deep wave tank, and aligned in parallel to the tank side walls (figure 3.1).

A thin gate separated water in the tank into two parts with different depths: 0.02 m deep on the square column side and 0.25 m deep on the other side. When the gate was lifted to the top of

the tank in 0.2 s by a 6.4-cm diameter pneumatic piston, a bore formed and propagated toward the square column downstream. The square column with a 12×12 cm square-shaped cross section was placed 5.2 m downstream from the gate. To measure hydrodynamic forces, the column was supported from above and connected with a force sensor.

The 3D OpenFOAM model incorporated the column into the computational domain by simply cutting off a block of mesh of the same shape from the computational domain. The mesh was coarse far from the column (1 cm by 1 cm by 0.5 cm in the x, y, z directions where the z is the vertical direction) and was refined gradually to 0.125 cm by 0.125 cm by 0.0625 cm in the x, y, z directions near the column surface. These distances are evaluated to be 70, 70 and 35 respectively in terms of dimensionless wall distance defined as $y^+ = \frac{yu^*}{\nu}$ where y is the distance, u^* is the friction velocity, and ν is the kinematic fluid viscosity. The mesh was finer in the z directions to better capture the water surface. Forces on the column were obtained by integrating pressure and shear forces from fluid on the surface of the column.

Water levels at the center of footprint of the column (column are removed) were compared with available experimental data . Streamwise forces on the column were also compared with the experimental data. Figure 3.2 shows the comparison between cases with and without the turbulence model. It reveals that the turbulence model can improve the accuracy of the prediction: the oscillation in predicted water level and the overestimation in the peak force were eliminated by using a turbulence model that damps out some of the flow features by including the eddy viscosity.

Assuming incompressible flow, the corresponding RANS equations can be written as:

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0 \tag{3.1}$$

$$\rho \frac{\partial \overline{u}_i}{\partial t} + \rho \overline{u}_j \frac{\partial \overline{u}_i}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \mu \frac{\partial^2 \overline{u}_i}{\partial x_j \partial x_j} - \frac{\partial \rho \overline{u_i' u_j'}}{\partial x_j} \tag{3.2}$$

where $\overline{u}_i$ is the mean velocity in the i direction, $u_i{}'$ is the fluctuating component of velocity in the i direction and $\overline{p}$ is the mean pressure. If u_i is the velocity component in the i direction, then

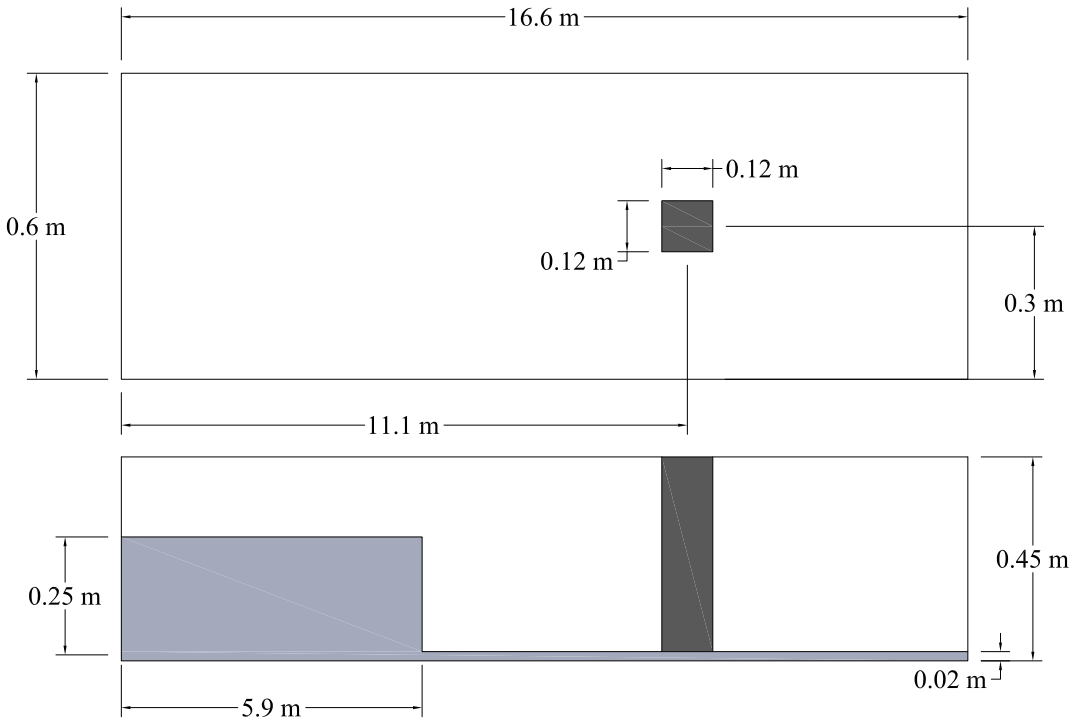
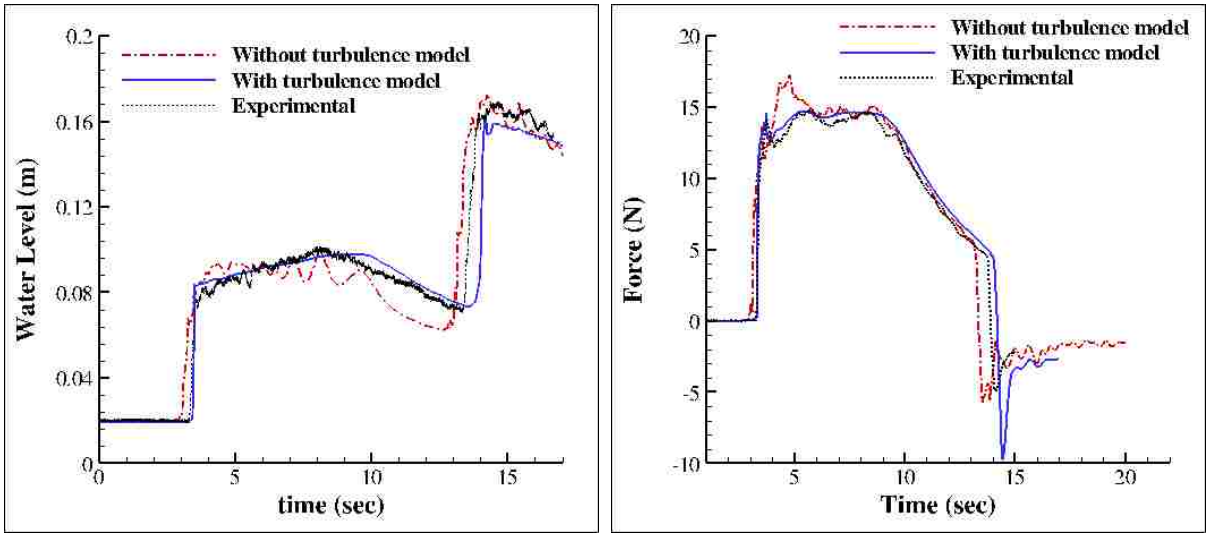


Figure 3.1: Schematic of the experimental setup for the interaction between bore and square column. The top figure shows a plan view, and the bottom figure shows a cross section through the center of the column, illustrating also the bore.



(a) Water level

(b) Streamwise forces

Figure 3.2: Comparison between cases with and without a turbulence model

$u_i = \overline{u}_i + u_i'$. The Reynolds Stress term in equation (3.2) is computed with:

$$-\rho\overline{u_i' u_j'} = \nu_t \rho \left[\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right] - \frac{2}{3} k \rho \delta_{ij} \tag{3.3}$$

where k is the turbulence kinetic energy and ν_t is the turbulence eddy viscosity. To close the equations above, Menter's k-ω-SST model [Menter and Esch, 2001] was applied to solve for k and the specific dissipation rate, ω:

$$\frac{\partial k}{\partial t} + \nabla \cdot (\mathbf{U} k) = \widetilde{G} - \beta^* k\omega + \nabla \cdot \left[(\nu + \alpha_k \nu_t) \nabla k\right] \tag{3.4}$$

$$\frac{\partial \omega}{\partial t} + \nabla \cdot (\mathbf{U} \omega) = \gamma S^2 - \beta \omega^2 + \nabla \cdot \left[(\nu + \alpha_\omega \nu_t) \nabla \omega\right] + (1 - F_1) CD_{k\omega} \tag{3.5}$$

where $\widetilde{G}$ is defined as $\widetilde{G} = \min\{G, c_1 \beta^* k\omega\}$, where G is the production term and defined as:

$$G = \nu_t S^2 \tag{3.6}$$

and S is invariant measure of the strain rate which is defined by:

$$S = \sqrt{2S_{ij}S_{ij}} \tag{3.7}$$

and S_{ij} is strain rate tensor defined by $S_{ij} = \frac{1}{2}\left(\nabla \mathbf{U} + \mathbf{U}^T\right)$. F_1 is a blending function defined by:

$$F_1 = \tanh\left\{\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\alpha_{\omega 2}k}{CD^*_{k\omega}y^2}\right]\right\}^4\right\} \tag{3.8}$$

where $CD^*_{k\omega}$ is defined by:

$$CD^*_{k\omega} = \max\left(CD_{k\omega}, 10^{-10}\right) \tag{3.9}$$

and $CD_{k\omega}$ is defined by:

$$CD_{k\omega} = 2\sigma_{\omega 2}\nabla k \cdot \frac{\nabla\omega}{\omega} \tag{3.10}$$

After solving equations (3.4) and (3.5), ν_t can be calculated by:

$$\nu_t = \frac{a_1 k}{\max\left(a_1\omega, SF_2\right)} \tag{3.11}$$

where F_2 is a second blending function defined as:

$$F_2 = \tanh\left\{\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right)\right]^2\right\} \tag{3.12}$$

All other constants are computed using a blend from the corresponding constants associated with the k-ϵ and k-ω models via blending functions like $\phi = \phi_1 F_1 + \phi_2\left(1 - F_1\right)$. Values for these constants are: $\alpha_{k1} = 0.85013, \alpha_{k2} = 1.0, \alpha_{\omega 1} = 0.5, \alpha_{\omega 2} = 0.85616, \beta_1 = 0.075, \beta_2 = 0.0828, \gamma_1 = 0.5532, \gamma_2 = 0.4403, \beta^* = 0.09, a_1 = 0.31, c_1 = 10.0$ [Menter et al., 2003].

It should be noted that the stability analysis of Mayer and Madsen [2001] showed that, for spatially large scale and low strain deformation fields, excessive unphysical production of turbulence could occur in waves if standard turbulence production terms are used. However, the rate of excessive unphysical production of turbulence in waves is slow. In their paper, a case with periodic cnoidal waves was tested and the turbulence did not get overestimated even after seven wave periods. In this study, the incoming wave can be treated as having only one wave period, which is not long enough for the turbulence to grow up to an unphysical state.

A force vector, $\mathbf{F}$, on an internal structural face is computed by summing forces from pressure, $\mathbf{F}_p$, and from viscous stress , $\mathbf{F}_v$.

$$\mathbf{F} = \mathbf{F}_p + \mathbf{F}_v \tag{3.13}$$

$\mathbf{F}_p$ and $\mathbf{F}_v$ are calculated respectively by:

$$\mathbf{F}_p = \sum_i \left(-p_i A_i \mathbf{n_i}\right) \tag{3.14}$$

$$\mathbf{F}_v = \sum_i \left\{\left(\tau_\mathbf{i} \cdot \mathbf{n_i}\right) A_i\right\} \tag{3.15}$$

where i denotes index of cell faces on the patch on which forces needed to be evaluated, p_i is total pressure on face i, A_i is area of face i, $\mathbf{n_i}$ is a unit normal vector of face i pointing into the computational domain and $\tau_\mathbf{i}$ is viscous stress tensor at face i which can be expressed by $\tau_\mathbf{i} = \left\{\rho\left(\nu + \nu_t\right)\left[\nabla\mathbf{U} + \nabla\mathbf{U}^T\right]\right\}$ on face i. For the forces presented herein, viscous forces are essentially negligible relative to the pressure forces.

Table 3.1 lists boundary conditions for each boundary in the numerical wave basin, which was developed to reproduce the physical experiment described in detail in Section 3.2. It is 48.8 m long and 2.1 m deep but narrower in width since only subsections of the experimental basin were modeled. Its bottom, side walls, two end walls and surfaces of internal structures were grouped as Walls in the table; upper boundary was named Atmosphere in the table. A *zeroGradient* boundary condition specifies that the normal gradient of a certain field quantity on a boundary face is zero: $\frac{\partial\phi}{\partial n} = 0$ where ϕ is the quantity on the boundary (the same for all ϕ hereafter in this section) and n is a unit normal vector of the wall. For an *inletOutlet* boundary condition, a *zeroGradient* boundary condition is used for outflow (when the velocity vector next to the boundary points outside). For inflow (when the velocity vector aims inside), it switches to a *fixedValue* boundary condition: $\phi = c$ where c is a constant value specified by the user. For α_{water}, this constant value is set to 0 since inflow on upper boundary should not consist of any watet. For the velocity boundary conditions, the *fixedValue* condition for the velocity field allows for implementation of a no-slip condition on all physical walls of the basin and internal structures by defining the velocity field on those

faces to be zero uniformly at any location along the boundary. The *pressureInletOutletVelocity* condition on the top of the domain is a *zeroGradient* boundary condition at all times except that the tangential component of the velocity can be set to *fixedValue* for inflow. In this case, this tangential component is set to 0, which makes this *pressureInletOutletVelocity* essentially a *zeroGradient* boundary condition. For the Walls and the internal structures, p_{rgh} (pressure subtracted by static pressure $\rho g h$ where ρ is the water density, g is the gravitational acceleration and h is relative depth under initial free surface) was defined such that there is zero flux using the *fixedFluxPressure* boundary condition (on a solid wall, this essentially turns to *zeroGradient*), while the Atmosphere was defined with a uniform reference pressure p_0 using the *totalPressure* boundary condition:

$$p_{rgh} = \begin{cases} p_0 & \text{, for outflow} \\ p_0 - \frac{1}{2}|\mathbf{U}|^2 & \text{, for inflow} \end{cases} \tag{3.16}$$

Internally, initial values for α_{water} were defined within every cell to specify whether the cell contains water or air, while $\mathbf{U}$ and p_{rgh} were zero because the flow was initially at rest.

Near the solid wall boundary, wall functions are applied, which allows evaluating the values of the turbulent quantities as functions of distance from the boundary. The size of the first layer of cells to the wall is chosen such that their centers are located in the log-law region of the boundary layer and corresponding wall functions can be applied. A *kqRWallFunction* boundary condition specifies $\frac{\partial k}{\partial n} = 0$ for k on a wall boundary where n is a unit normal vector of that wall. An *omegaWallFunction* boundary condition provides a wall function for the turbulence specific dissipation ω. It is computed with:

$$\omega = \sqrt{\omega_{vis}^2 + \omega_{log}^2} \tag{3.17}$$

where ω_{vis} is value of ω in viscous region and ω_{log} is value of ω in logarithmic region [Menter and Esch, 2001]. A *nutUSpaldingWallFunction* boundary condition is a boundary condition for ν_t when wall functions are used for rough walls. It uses Spalding's law [Spalding, 1961] to compute a continuous ν_t profile to the wall. Spalding's law is essentially a unified law of the wall which can fit the viscous sublayer, buffer layer and the logarithmic region in a boundary layer:

$$y^+ = u^+ + \frac{1}{E}\left[e^{\kappa u^+} - 1 - \kappa u^+ - 0.5(\kappa u^+)^2 - \frac{1}{6}(\kappa u^+)^3\right] \tag{3.18}$$

Table 3.1: OpenFOAM boundary conditions for the current numerical model

Field	All walls and floor	Atmosphere
Air/water phase indicator, α_{water}	zeroGradient	inletOutlet
Velocity, **U**	fixedValue	pressureInletOutletVelocity
Pressure without hydrostatic part, p_{rgh}	fixedFluxPressure	totalPressure
Turbulent kinetic energy, k	kqRWallFunction	inletOutlet
Specific dissipation rate, ω	omegaWallFunction	inletOutlet
Turbulence eddy viscosity, ν_t	nutUSpaldingWallFunction	zeroGradient

where $\kappa = 0.42$ and $E = 0.91$ are constants, $y^+ = \frac{yu_\tau}{\nu}$, and $u^+ = \frac{u}{u_\tau}$. Here $u_\tau = \sqrt{\frac{\tau_w}{\rho}}$ where ρ is the fluid density and τ_w is the wall shear stress. Equation (3.18) can be used to compute the wall shear stress with a iterative procedure and ν_t subsequently.

The initial condition is set such that the flow is still at the beginning. Thus the velocity is set to zero in the whole domain. The seeding value for turbulence kinetic energy, k, is computed by

$$k = \frac{1}{2}(u_1'^2 + u_2'^2 + u_3'^2) \approx \frac{1}{2}u_1'^2 \tag{3.19}$$

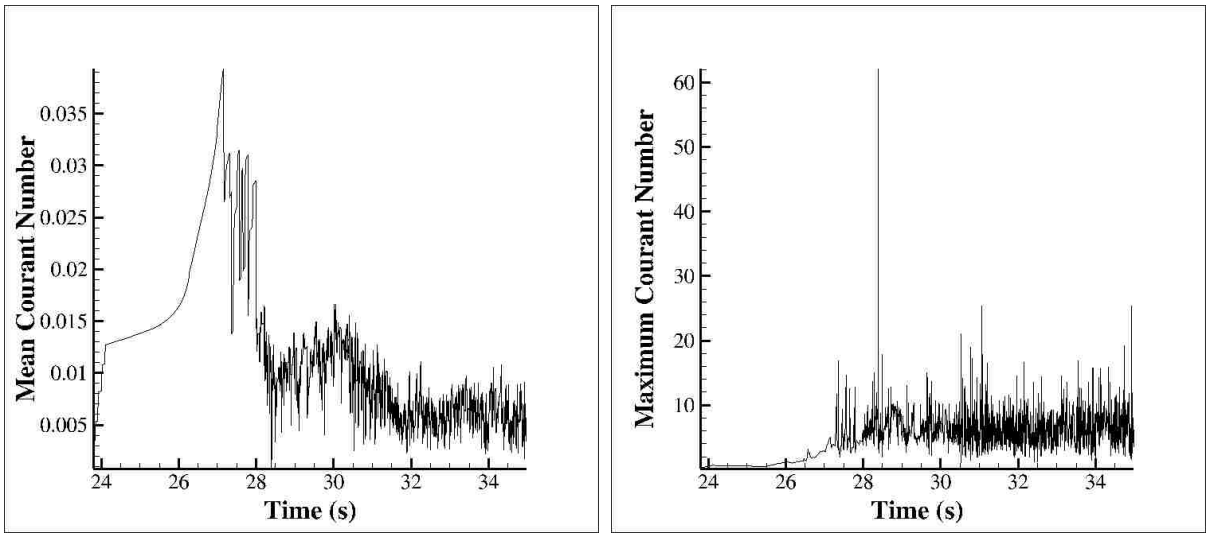
with assumption of zero velocity fluctuation in the along-shore and vertical direction. The velocity fluctuation u_1' is computed from $I = \frac{u_1'}{U}$ where I is turbulence intensity and U can be chosen as wave celerity in this case, which is the same as Svendsen [1987] and Lin and Liu [1998]. The initial turbulence intensity is chosen to be 1% in this simulation. We also assume that the tsunami wave has propagated for a long distance with water depth unchanged before it enteres into the computational domain. Thus the flow is assumed to be stationary at the inlet, which makes us use $\frac{\sqrt{k}}{l}$ for the specific dissipation rate, ω, where l is the turbulent length scale and is set to 7% of the hydraulic diameter of the channel-like computational domain, according to Pope [2001].

To achieve numerical stability, a time step, δt, is automatically computed by the solver such

that the Courant number ($\delta t\,|\mathbf{U}|/\delta x$) is less than a user-defined maximum value of 10 in all cells, where $\mathbf{U}$ is the velocity vector through the cell and δx is the size of the cell in the direction of the velocity. In the version of OpenFOAM used in this dissertation (version 2.3.1) [The OpenFOAM Foundation, 2014b], a new semi-implicit variant of the previously used multi-dimensional limiter for explicit solution (MULES) [The OpenFOAM Foundation, 2014c] is introduced which first computes a predictor step based on purely bounded implicit methods and after that, constructs an explicit correction with MULES limiter applied. This implicit predictor step ensures boundedness and stability at an arbitrarily large Courant number. Although the correction is updated and applied frequently to ensure accuracy, the semi-implicit approach is overall substantially faster than the explicit method in previous version of OpenFOAM [The OpenFOAM Foundation, 2014a] due to the limit on Courant number. It has been shown that for large scale simulations like ship-keeping simulations, this new method reduces run times significantly [Moctar et al., 2012].

Since the Courant number can be arbitrarily large while boundness and stability can be maintained in this version, several Courant number setups in OpenFOAM were tested in order to maximize the Courant number (and thus decrease computational time) while maintain accuracy. It is found that maximum Courant numbers set to 10 increased computational time but did not produce a better result, while increasing the Courant number beyond 10 led to lower quality results. Figure 3.4 shows results where the maximum allowed Courant number for interFoam is set to 10, 5, 2.5 and 1 respectively. Differences in water level between different cases are small, given the uncertainties in this problem.

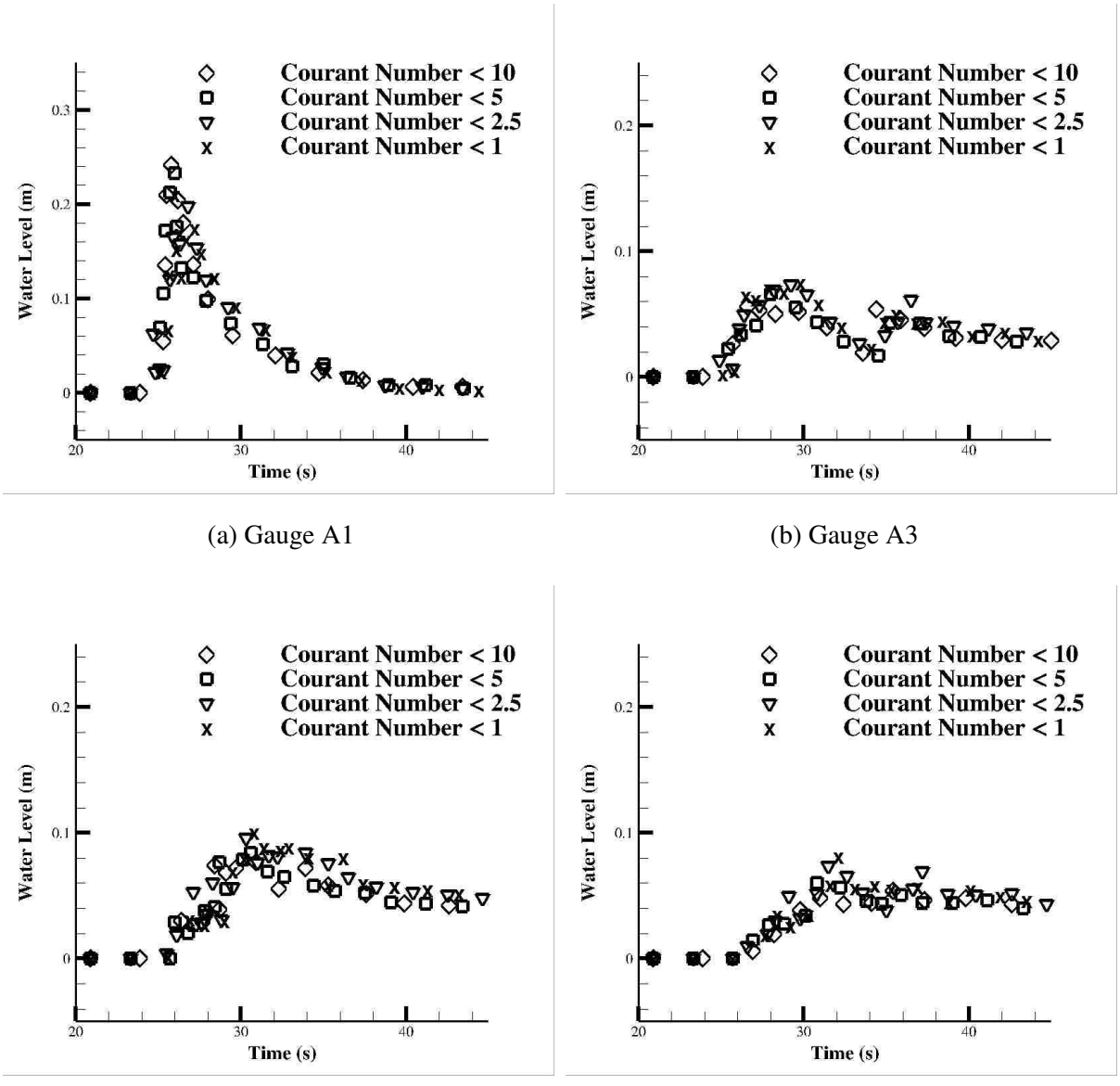
Figure 3.3 shows the actual mean Courant number and maximum Courant number during a sample simulation for the Seaside model described later in this chapter. It shows that even with "maximum courant number" set to 10 in OpenFOAM, the maximum Courant number can still exceed that at some moments. These high Courant numbers occur in the smallest mesh cells. The reason for this is explained at the end of section 3.2.1. In other much bigger mesh cells, the Courant numbers are all much smaller. This can be verified by looking at the "mean Courant number", which is never larger than 0.04 during the simulation and indicates that the Courant numbers in nearly all of the mesh cells are moderate.



(a) Mean Courant Number

(b) Maximum Courant Number

Figure 3.3: Mean and Maximum Courant Number in the case with "Maximum Courant Number" set to 10



(a) Gauge A1

(b) Gauge A3

(c) Gauge A6

(d) Gauge A7

Figure 3.4: Surface elevation at gauges A1, A3, A6 and A7 for cases with different maximum Courant numbers

3.2 Model Setups and Validation

The experiment on tsunami inundation through an urban waterfront that was used here to compare to the predicted flow characteristics of the 3D model was conducted in the Tsunami Wave Basin at the O.H. Hinsdale Wave Research Laboratory at Oregon State University. A 1:50 scale physical model of part of the town of Seaside, Oregon, located on the U.S. Pacific Northwest coastline and adjacent to the Cascadia Subduction Zone (CSZ), was constructed and a series of experiments were conducted to measure flow velocities and the water levels at 31 locations within the model-scale community. Full details of the experiment can be found in Park et al. [2013]. For completeness, some of those details are presented here.

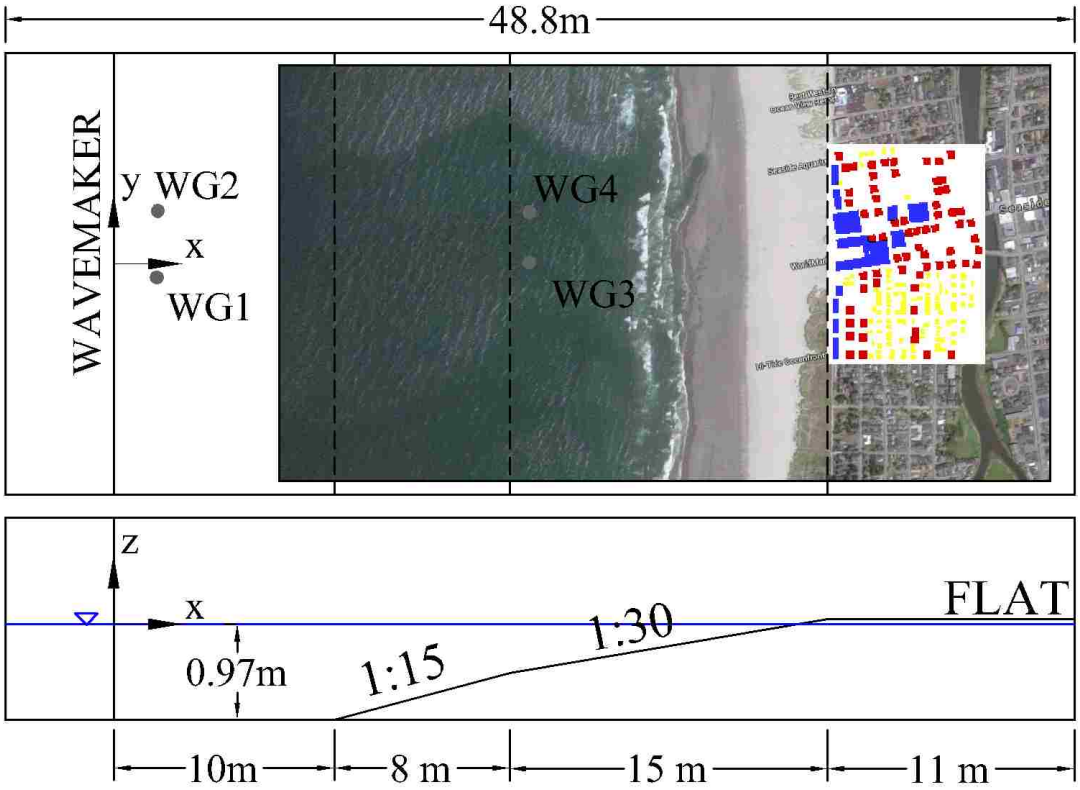


Figure 3.5: Top view and side view of the basin

The rectangular basin is 48.8 m long, 26.5 m wide and 2.1 m deep and was equipped with a

segmented, piston-type wavemaker with a maximum stroke of 2.1 m and maximum velocity of 2.0 m/s Cox et al. [2008]. Figure 3.5 shows top view and side view of the basin, which has a 10 m horizontal section starting from wavemaker, followed by an 8 m section at a 1:15 slope and after that, a 15 m section at a 1:30 slope. Another flat horizontal section, 11 m in length, connects the 1:30 slope section to the back wall of the basin. The initial still water depth near the wavemaker is 0.97 m and decreases as it approaches the shoreline, where the still water line intersects with the 1:30 slope. All idealized buildings in the town were fixed on the upper horizontal section. In the front of the town, there was a 0.04 m high (model scale) seawall (the hump showed in the right half of figure 3.6) parallel to and 32 m away from the wave maker [Park et al., 2013].

Figure 3.6 and 3.7 show more details of the buildings onshore and the locations of the 31 gauges where water level and flow velocity were measured in the experiment. The gauges are grouped into 4 groups, group A, B, C and D in figure 3.7. Buildings are categorized into three types: large commercial building (like hotels and hospitals) colored by blue, small commercial building colored by red and residential structures colored by yellow. In this experiment, buildings in red and yellow all have the same dimensions while those in blue do not. The right half of figure 3.6 actually shows the geometry file in STL (STereoLithography) format used in the current 3D numerical model, which is read by a mesh tool, snappyHexMesh, to generate meshes used by OpenFOAM. This geometry file was built in a 3D modeling software after measuring dimensions and locations of the structures from given point cloud data, which was directly recorded by scanning the surfaces of the physical model in the laboratory. The geometry of all structures were put in one global coordinate to ensure the correct relative locations.

To simulate tsunami impact, the piston motion was designed to provide, at the lower horizontal section of the basin, an initial wave with a wave height of approximately 0.2 m, which is equivalent to 10 m at prototype scale. This corresponds to a 500-year CSZ tsunami for this region [Tsunami Pilot Study Working Group and others, 2006]. With identical incoming waves, multiple trials were performed and data were measured during each trial. Due to stochastic characteristics in the experiment, the measurements varied between individual trials and ensemble averaged values were used to describe the result, details of which were presented in Park et al. [2013].

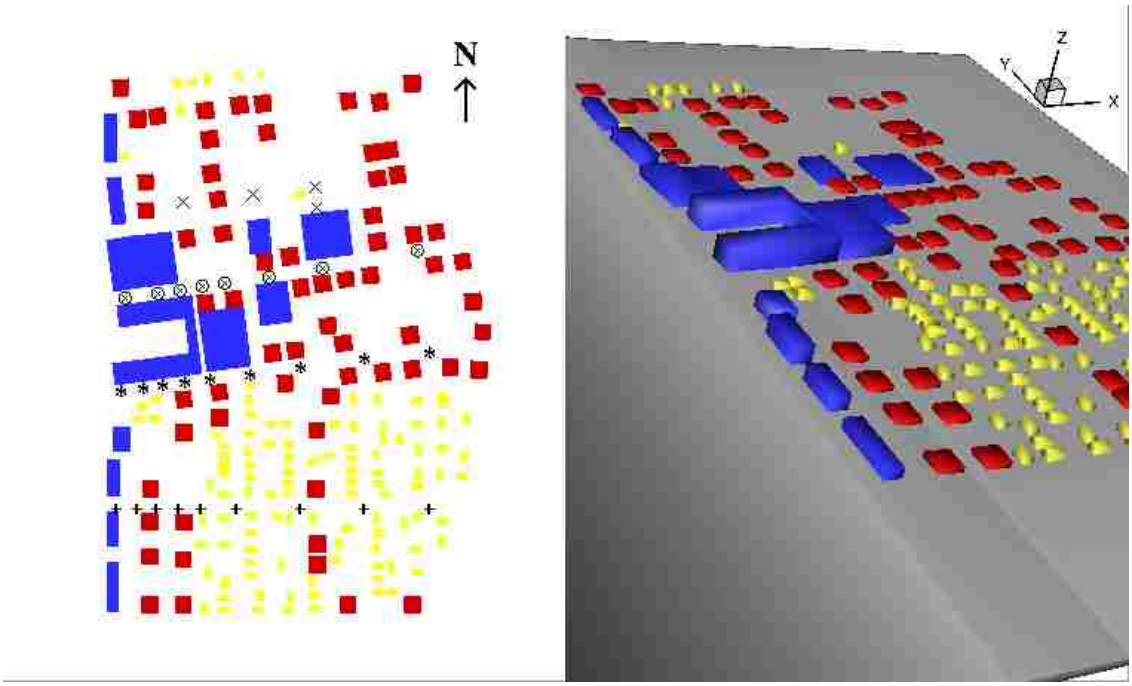


Figure 3.6: Layout of all buildings and gauges in the experiment: blue, large hotels or commercial buildings, red, smaller commercial buildings, yellow, residential structures.

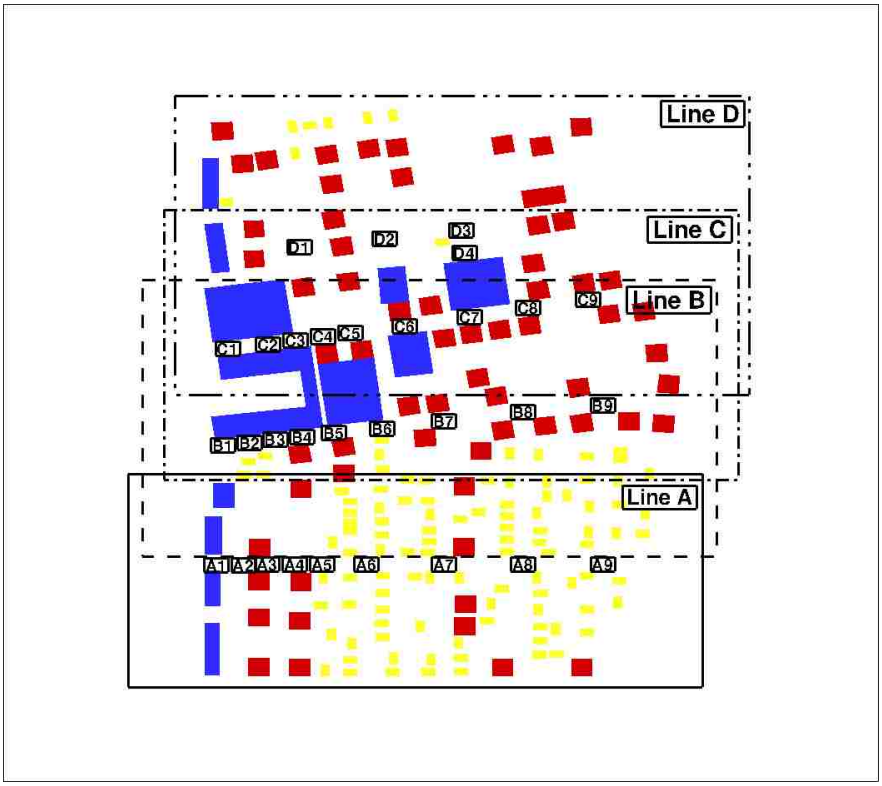


Figure 3.7: Layout of gauges and their names. The four overlapping rectangles indicate four subsections used to get the numerical results at gauge group A, B, C and D. For clarity, only the width of the domain in the vicinity of the onshore bathymetry is shown in the figure; however, the numerical domain spans the 48.8 m from the wavemaker to the back wall of the basin.

Using OpenFOAM, a 3D numerical model was developed to simulate the experiment. It was built at the same scale as the experiment instead of the prototype scale to exclude scaling effects so that a better comparison with the experimental data can be achieved. To generate numerical tsunami waves, a numerical wave flume was previously developed in OpenFOAM [Motley et al., 2014] and it was validated against available data from a few experiments. The numerical wave flume simulated waves generated by a piston-type wave maker using a two-step process that took advantage of the fact that the motion of the wave maker occurred over only a short period of time and that the tank directly in front of the wave maker has a flat bottom and is free of obstructions. First, a short extra chunk was added to the original domain ahead of the wave maker while the wave maker was moving. This step was conducted using the wave maker as the reference frame, eliminating the need for a moving mesh, and fluid was allowed to enter the domain at wave maker's speed at the end of the domain opposite to the wave maker. Since this was a non-inertial frame, a time-varying acceleration vector field was also embedded into the solver. The second step began when the wave maker stopped moving and the wave had passed the position of wave maker. All field data in the domain from the first step was mapped to a new mesh of the basin or water tank with the mapFields utility in OpenFOAM and the simulation could continue from there.

3.2.1 Selection of the Computational Domain and Mesh Resolution

One disadvantage of the 3D model is that it is much more computationally expensive compared to a typical 2D model. The wave basin could not be modeled as a whole even with all available computational resources for this study. Instead, the entire domain was divided into four subsections to predict flow parameters at four groups of gauges. One question that follows is then how wide the subsection needs to be in order to minimize the influence of numerical boundary on both sides of the subsections, while keeping the required computational resources in a practical range.

To answer this question, two numerical domains with different widths were first constructed and they were both chosen to include the gauges along Line A (See figure 3.8). For clarity, only the width of the domain in the vicinity of the onshore bathymetry is shown in the figure; however, the numerical domain spans the 48.8 m from the wavemaker to the back wall of the basin. The

two domains that were modeled were 2.3 m and 5.0 m wide. The narrower case (2.3 m x 48.8 m) required a computational mesh with approximately 25 million cells and took approximately 3 days to compute on 4 dual 8-core 2-GHz Intel Xeon e5-2650 machines (64 total processors) in parallel. The wider case (5.0 m x 48.8 m) required a computational mesh with approximately 60 million cells and took 9-10 days on the same machines.

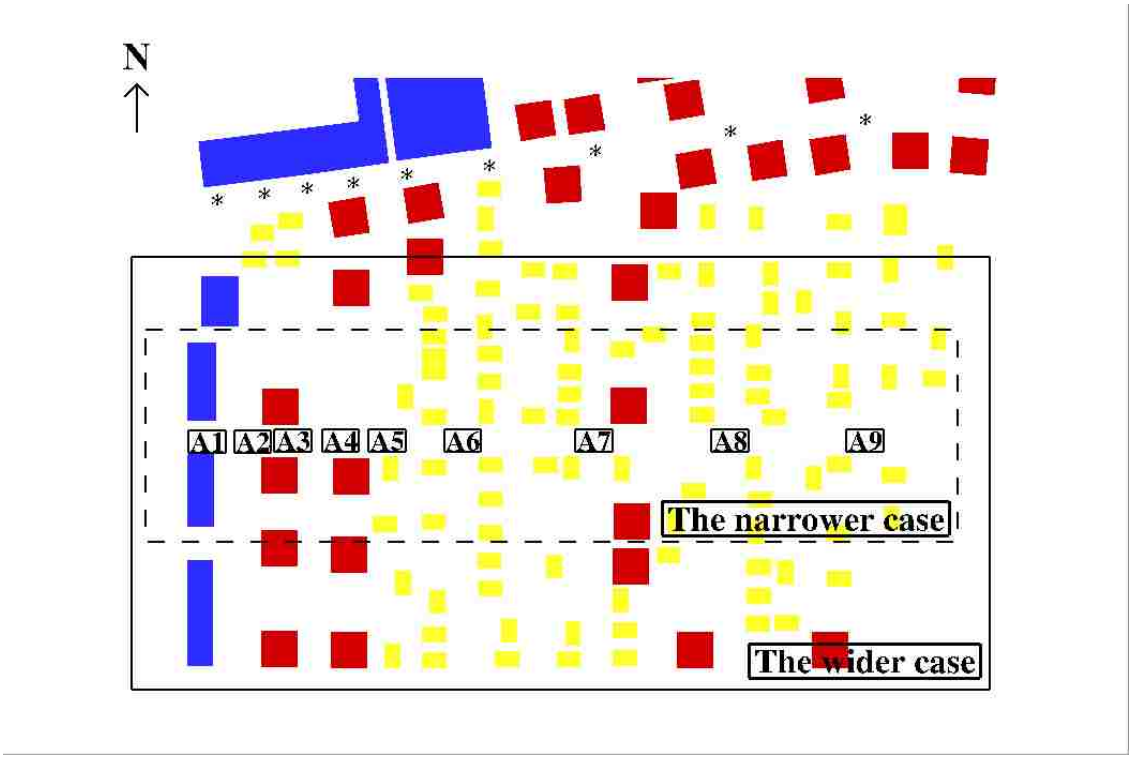


Figure 3.8: Computational domain for the narrower case and the wider case For clarity, only the width of the domain in the vicinity of the onshore bathymetry is shown in the figure; however, the numerical domain spans the 48.8 m from the wavemaker to the back wall of the basin.

During the simulation, there were two distinct modeling phases with critical mesh refinement focused on different regions to mitigate computational demand. The first phase was from the beginning of the simulation to the time when wave had passed the offshore gauge WG3 and almost started to break. In this phase, all buildings were removed from the domain, leaving only the flat

bottom of the wave basin, which reduced number of cells needed onshore significantly, allowing for use of a much finer mesh offshore. During this phase, the mesh size was approximately 0.08 m $\times$ 0.08 m $\times$ 0.01 m (length $\times$ width $\times$ height) near the wave maker and was gradually reduced to 0.08 m $\times$ 0.08 m $\times$ 0.004 m onshore due to changes in topography. In the second phase, buildings were reintroduced into the domain and the offshore mesh was coarsened to allow for a finer mesh near the onshore bathymetry. The mesh size for the second phase was 0.3 m $\times$ 0.015 m $\times$ 0.035 m near wave maker and refined to 0.0075 m $\times$ 0.0075 m $\times$ 0.0025 m near the flat bottom of the onshore segment of the basin and at the edges and corners of the buildings. Simulation results from the end of the first phase were mapped to the second phase and the simulation continued. This strategy is similar to the adaptive mesh refinement algorithm that is used in the 2D Seaside models described in chapter 4. Here, however, a statically refined mesh was used instead of an adaptively refined mesh.

Before the mesh size above was chosen, different sizes of mesh cells were tested and the final meshes were chosen to achieve a balance between accuracy and computational time. Result of a convergence test conducted with the narrower domain is shown in figure 3.9. Three cases with different mesh sizes (referred to as coarse, medium and fine) were tested. The mesh size of the fine mesh is the same as the mesh discussed above while sizes of the medium mesh and the coarse mesh are increased by a factor of 1.5 and 4 respectively from the fine case. Total number of cells for the coarse, medium and fine cases were 0.4 million, 6 million and 25 million respectively (for the second phase mentioned above).

It is worth noting that there were much smaller cells around the buildings than those described above. All mesh sizes described above were the sizes of structured background mesh that would be processed by the unstructured mesh generator, snappyHexMesh, which would further divide these background mesh cells into smaller ones where they intersected with the boundaries of domain (side walls and buildings etc.). In the current study, mesh cells that intersected with the boundaries were further divided by a half in each direction (1 cell becomes 8 smaller ones). After this, a less controllable process was conducted which moved grid points near an object surface (boundary) to make those points be exactly on the surface. This could generate highly skewed and very small

mesh cells near the edges and corners of the buildings. As a result, the ratio of size of the largest cell to that of the smallest cell could be as large as in the order of 10^5 in the current models.

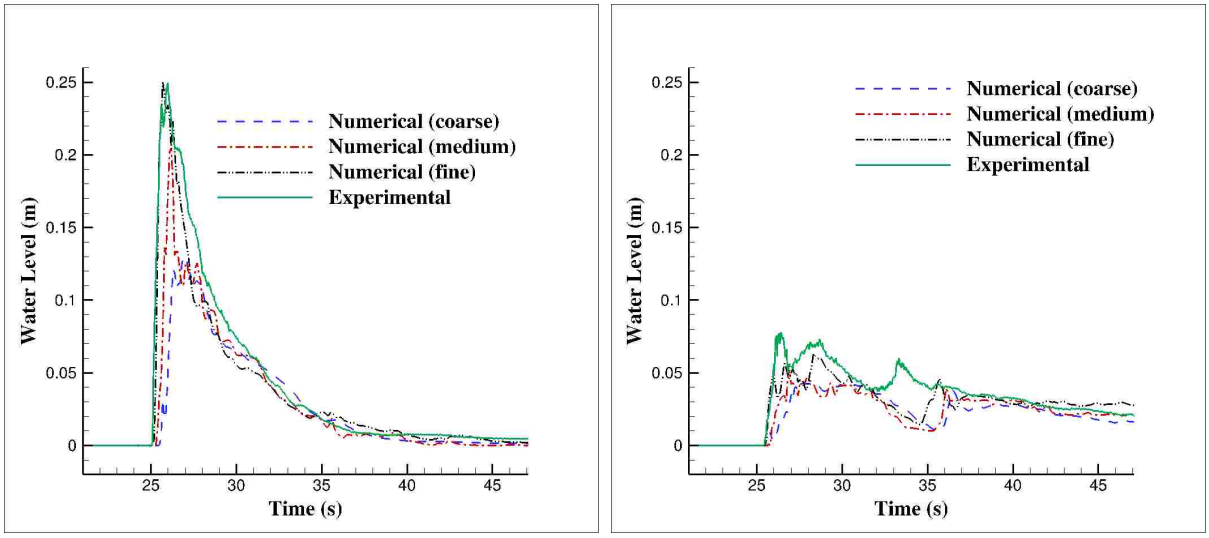
Due to the rapid increase of computational resources required as the mesh size decreased, the mesh in the fine case was the finest mesh tested in this convergence test. The convergence test showed that further mesh refinement would not improve the accuracy sufficiently to justify the expense of much larger (and approaching impractical) computational times based on available resources.

In the 3D Seaside model used for getting all numerical results for the rest of this dissertation, the mesh from the fine case was chosen. The aspect ratio of mesh near wavemaker is quite high, but that was found to have no influence on wave generation and propagation offshore from several initial trials with varying aspect ratios. The aspect ratio used here was selected to minimize computational time while maintaining accuracy.

3.2.2 Comparison of Flow Parameters

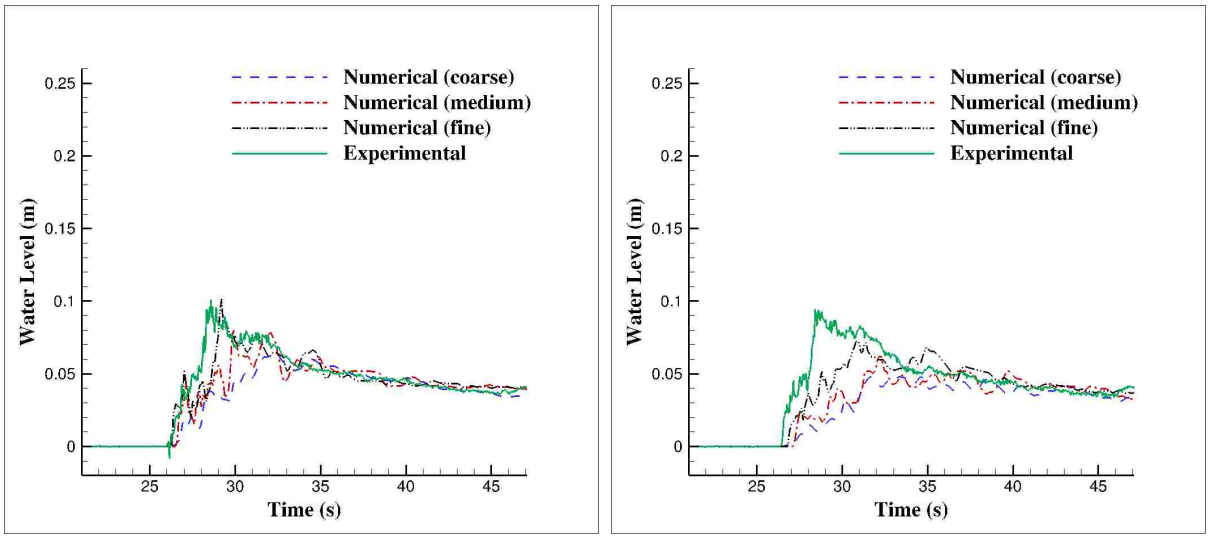
The predicted free surface elevation (water level), cross-shore (x-component) velocity of the flow, and corresponding momentum flux ($M = hu^2$, where h is the free surface elevation and u is the cross-shore velocity) at selected group A gauges from the 3D OpenFOAM model is presented and discussed in this section. Note that this dissertation uses the term cross-shore to refer to the direction that the flow propagates from the wavemaker to the back wall, while the term along-shore is used to refer to flows traveling perpendicular to the cross-shore flow in the direction of the width of the domain.

Figure 3.10 shows the time history of the free surface elevation at two wave gauges offshore (see figure 3.5 for locations of the two gauges). Note that this single peak wave generated in the experiment is not a solitary wave but an impulse-type wave based on the error function [Rueben et al., 2011]. The wavemaker displacement time series from the experiment was used for the moving wall boundary condition in the current numerical model to generate the numerical waves. Although the tsunami wave generated in the numerical model has slightly lower wave height, narrower crest width, slightly slower propagating speed and a longer "tail" following the peak,



(a) Gauge A1

(b) Gauge A3



(c) Gauge A6

(d) Gauge A7

Figure 3.9: Surface elevation at gauge A1, A3, A6 and A7 for cases with different mesh size

correlation between the measured and predicted results were overall satisfactory.

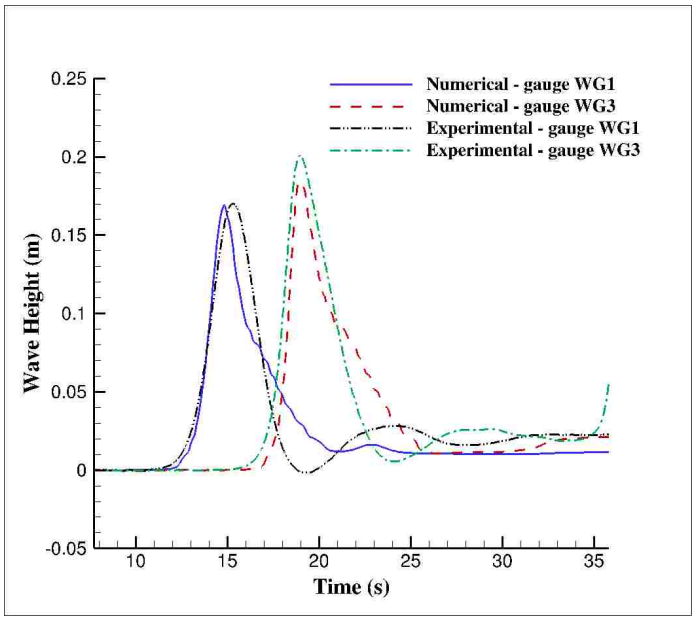


Figure 3.10: Time histories of surface elevation at gauge WG1 and WG3. Only numerical result from the wider case is plotted as the two cases gave the same result in this region.

Time histories of the free surface elevation, cross-shore velocity and corresponding momentum flux at gauge A1, A4, A6 and A7 are shown in figure 3.11. The first row in each of the subplots shows a comparison of time histories of the water level between the experimental data and the numerical results of both the narrower and wider models. Correlation between the numerical and experimental results are generally good, and there are several interesting results to note. At gauge A1, A4 and A6, both numerical cases showed very good agreement with experimental measurements, in terms of amplitude and arrival time. At gauge A7, the predicted water level from the wider case agreed better with experimental results than the narrower case. This is most likely because more water was able to enter the domain for the wider case, and the corresponding interactions between the wave inundation and constructed environments are much more complex as the wave moves farther onshore and water flows more and more in the along-shore direction. As a result, the wider case was chosen for further investigation of flow and forces near group A gauges.

Other gauges (gauges in group B,C and D) were also validated using different subsections (denoted by different type of lines in figure 3.7) and the corresponding results are shown in chapter 4 for better comparison against the results from 2D models.

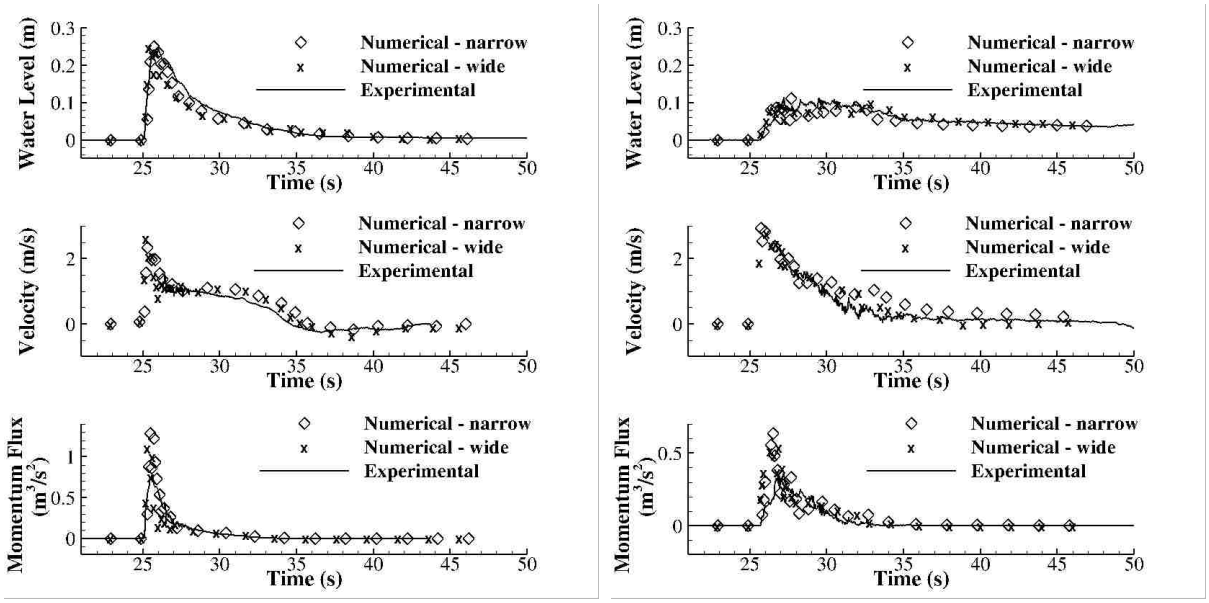
The second row in each subplot of figure 3.11 shows time histories of cross-shore velocity. In the experiment, the velocity at gauge A1 became negative after t = 35 seconds, which indicated a back flow from the seawall. The maximum absolute value of the velocity in this back flow rose up to 0.2 m/s (from onshore to offshore). This did not occur in numerical model before t = 40 seconds. The overlong period (wave length) of the numerical wave in the current model contributed to this delay (See figure 3.10). In the numerical model, the wave was tailed by an extra volume of water which formed a lower slope at wave back. This was not the case in the experiment, however, where the wave had a narrower shape and higher slope at the tail of the wave. This volume of water at the tail kept propagating above the seawall and pushing water onshore for some time after the initial arrival of the wave at gauge A1. For this reason the back flow in the numerical model did not occur at the same time as it did in experiment. In general, prediction of cross-shore velocity agrees well with the measurement except for time ranges near initial impact, where some experimental data are missing in these plots.

The last row in each subplot of figure 3.11 shows time histories of the momentum flux. As expected, the momentum flux agrees well with measurements where prediction of both water level and velocity are good, since the momentum flux is computed from these two quantities. However, the discrepancies in the momentum flux between prediction and measurement are quite large near the initial impact.

In general, predicted and measured flow parameters agree well, although velocities and momentum flux show large discrepancies near the peak, which may be due to measurement errors, as discussed in detail later.

3.2.3 Discrepancy in Velocity Prediction

Perhaps more significant discrepancies occur near the initial impact. Figure 3.12 shows time histories of flow parameters at the same gauges as in figure 3.11 but is zoomed in near initial impact.



(a) Gauge A1

(b) Gauge A4

(c) Gauge A6

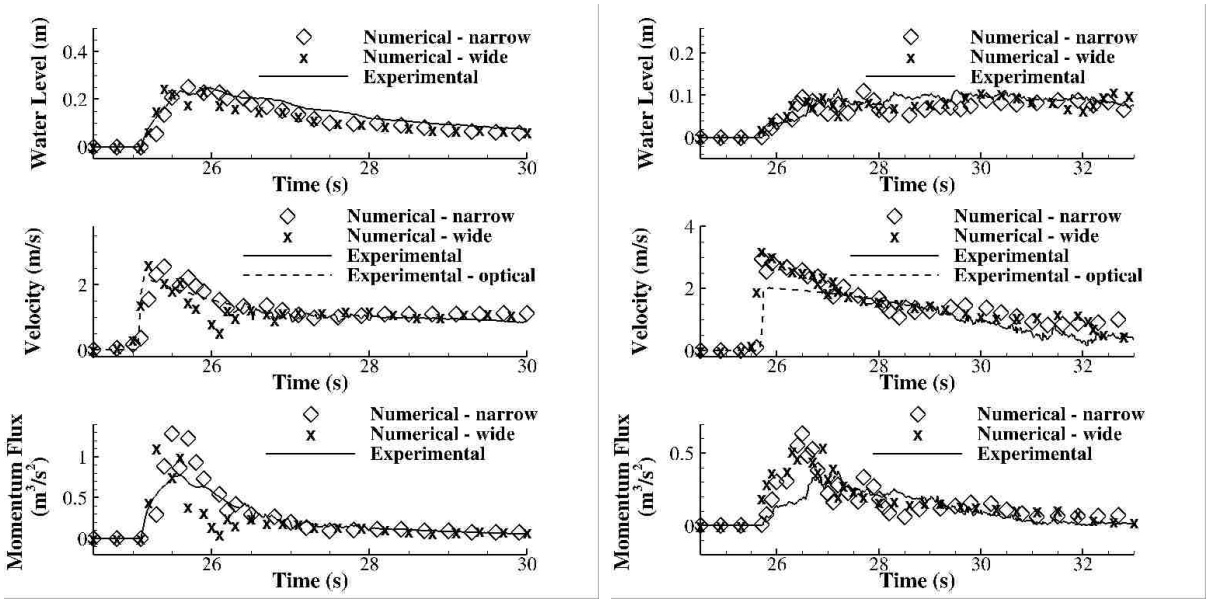
(d) Gauge A7

Figure 3.11: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges along line A (Note that ranges of Y axis are different in different subplots)

The dashed lines represent data obtained using optical measurements by Park et al. [2013]. In the experiment, when the bore front arrived, the Acoustic Doppler Velocimeter (ADV) failed to record velocity in some trials and could not give consistent measurements in others due to large amounts of air entrained inside the bore [Park et al., 2013]. Thus the peak value of velocity histories was not obtained from the ADV but from an optical measurement, where velocity was computed by analyzing the trajectory of the leading edge of the bore from image data taken by two high resolution video cameras located above the wave basin [Rueben et al., 2011]. A linear rise in velocity from initial ADV measurements to the predicted peak was applied, and a second-order polynomial approximation was then applied to connect the optically measured peak to available ADV measurements. In the current study, all experimental data were downloaded from the NTHMP Mapping and Modeling Benchmarking Workshop: Tsunami Currents [University of Southern California, 2015]. In these data, momentum flux near initial impact are available, which was used to reconstruct the velocities obtained by the optical approach in the experiment (dashed line in figure 3.12) with $u = \sqrt{\frac{hu^2}{h}}$.

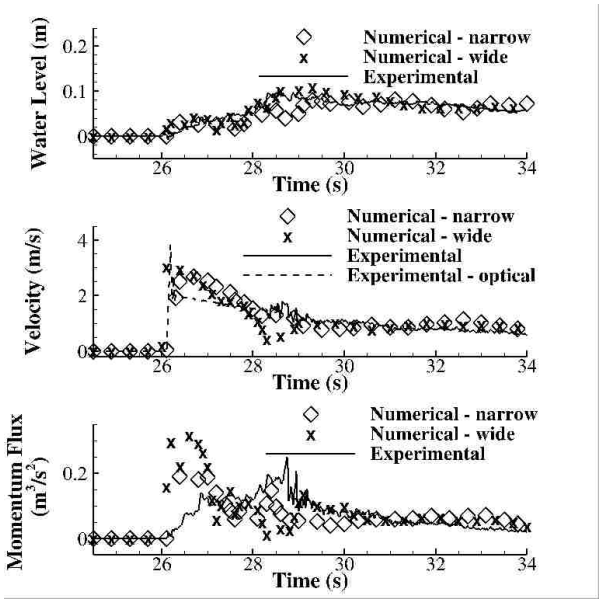
As shown in the figures, there are clear discrepancies in velocity predictions in the optically measured region between the experimental and numerical results. These discrepancies can be primarily attributed to the different measurement approaches. If an "optical" approach is used for the numerical results to compute the peak velocity in the numerical simulations, the predicted numerical results nearly match the experiment. Take for instance the measurement of velocity at gauge A4 in the narrower case as an example (figure 3.13). If location of the bore front at t=25.8 second was compared with the position at t=25.9 second, between which the bore front passed through gauge A4, the computed peak value of the velocity was $u = \frac{\triangle x}{\triangle t} \approx 2.2$ m/s at gauge A4, which agreed quite well with the optical measurement from the experiment (figure 3.12).

Note, however, that the contour profile of bore front in figure 3.13 is colored by cross-shore velocity. A closer check of the velocity field in the bore reveals that maximum velocity is around 2.8 m/s and it does not occur at the bore front but somewhere after that. This is why the numerical models show a peak value of 2.8 m/s in the time history of velocity for gauge A4 (figure 3.12). Thus if it is assumed that the peak value occurs at bore front, as is assumed in the experiment, the

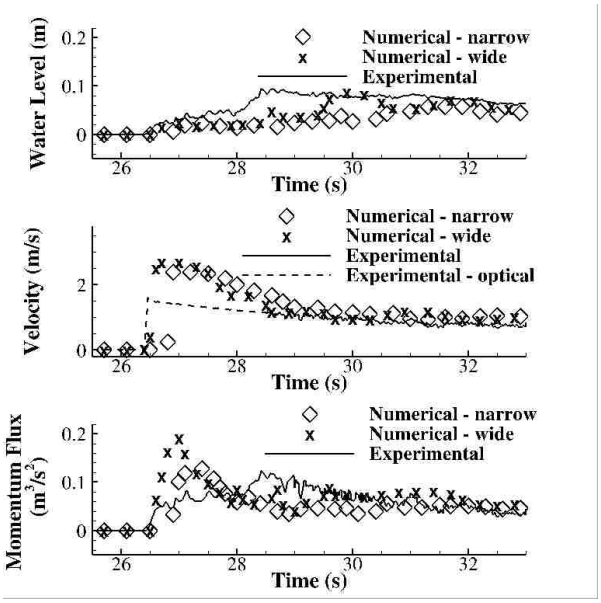


(a) Gauge A1

(b) Gauge A4

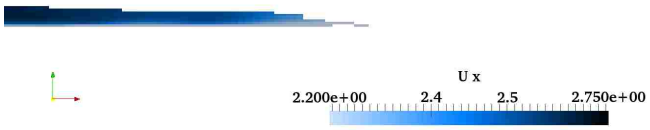


(c) Gauge A6

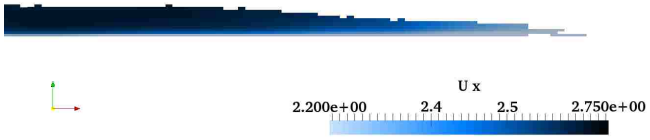


(d) Gauge A7

Figure 3.12: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges along line A, zoomed in near the initial impact



(a) $t = 25.8$ s



(b) $t = 25.9$ s

Figure 3.13: Position of bore front when peak value of velocity occurs at gauge A4, colored by velocity

peak value could be significantly underestimated. This becomes problematic if we predict dynamic fluid forces from the momentum flux, $M = hu^2$, where h is the depth of the water and u is the flow velocity.

Because of the square term, it is reasonable to expect that in many cases, the velocity will control the momentum flux calculation, which is shown explicitly in the momentum flux subplots (figure 3.12). The maximum momentum flux from both narrow and wide cases was found to be about two times higher than the experimental measurement. The water level from numerical models and experimental measurement agree quite well. Thus the over-prediction of the momentum flux is primarily due to the overestimation of velocity around the peak.

3.3 Force Predictions on Selected Buildings

The 3D Seaside model was described and validated against experimental data of flow parameters in section 3.2. Although wave forces on buildings are probably even more important information, they were not measured in the physical experiment. This section selects six representative buildings

along Line A for preliminary analysis of fluid forces on the coastal infrastructure based on the numerical model, as shown in figure 3.14. Buildings I and II are the two large structures located directly on the coast adjacent to gauge A1, with dimensions of 0.29 m by 0.78 m by 0.246 m (length in cross-shore direction, length in along-shore direction and height respectively, and the same for the following) and 0.31 m by 0.84 m by 0.31 m, respectively. Buildings III and IV are identical, with dimensions of 0.39 m by 0.39 m by 0.091 m. Buildings V and VI, representing houses within the community, are also identical, whose length, width and height are 0.17 m, 0.26 m and 0.154 m.

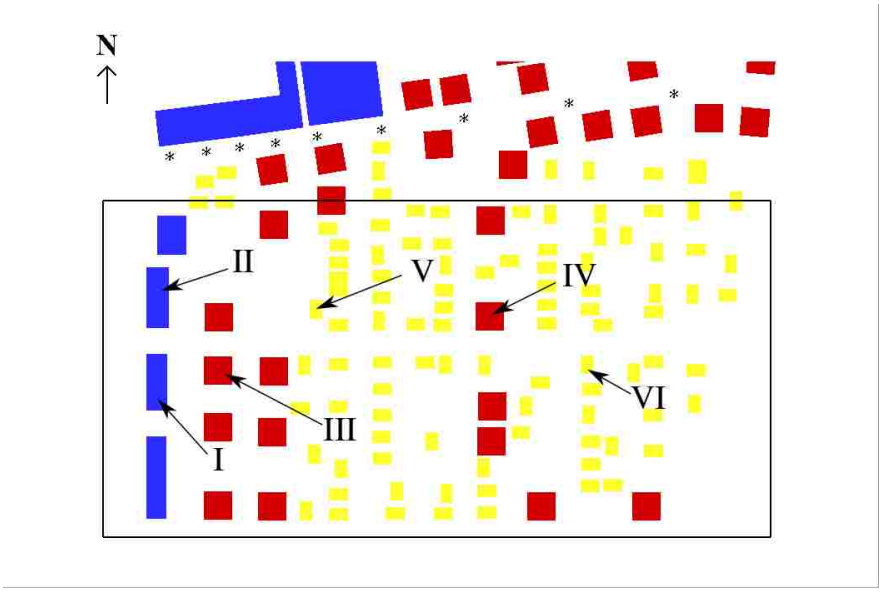


Figure 3.14: Representative buildings along Line A

Figure. 3.15 and 3.16 show predicted tsunami forces on these selected buildings in the cross-shore and along-shore direction. Note that in these figures, forces in cross-shore direction were normalized by the width of western left wall of the buildings, forces in along-shore direction were normalized by the width of northern wall of the buildings.

As shown in figure 3.15, Buildings I and II suffered from strong initial impact from tsunami waves. The peak normalized forces on Building I in the cross-shore direction reached approxi-

mately 570 N/m, which gives us 445 N in model scale. This is equivalent to a force of 55,600 kN in the prototype. Using the Equivalent Lateral Force procedure described in ASCE 7-10 and shown in Appendix A, the seismic load due to the Maximum Considered Earthquake, MCE, for an elastic structure (Response Modification Coefficient, R, equal to 1) is computed as 24,200 kN, or approximately 44% of the tsunami induced load. While the seismic design load for a structure expected to yield under the MCE (R equal to 5) was estimated as 3,350 kN, or approximately 6% of the tsunami induced load. While these tsunami loads are more than likely conservative estimates based on the fact that portions of a building may fail or that there may be non-structural components that would not be designed to resist load, this comparison shows that tsunami loads can be of the same order of magnitude as seismic demands.

As the water flows inland, the cross-shore components of forces on buildings greatly decrease. Even for Building III, located directly behind Building I, the impact load was approximately 20% as large as the impact for Building I, which served to shield Building III from much of the impact. Forces on buildings IV, V, and VI are substantially lower because the flow becomes shallower and slower as it interacts with the buildings.

In addition to these results, time histories of normalized forces in the along-shore direction, perpendicular to the cross-shore flow, are shown in figure 3.16. For Buildings I and II, which were impacted directly in the cross-shore direction, the magnitude of the along-shore forces are approximately 15% of the cross-shore forces. As the flow moves onshore and interacts with constructed environments, however, this is not the case. The peak force in the along-shore direction on building III, for instance, is as high as approximately 50% of the cross-shore loading. It is also interesting to note that the along-shore forces for several buildings fluctuate around zero, indicating turbulent, moving water on both sides of the structure.

To help understand the localized flows, figure 3.17 shows the horizontal angle from the cross-shore direction of net horizontal force profile on these selected buildings, where zero represents the cross-shore direction and angles increase counter-clockwise within the horizontal plane. Note that in this figure, data at the time when the cross-shore component of the net force was negative (pointing offshore) or when net force at this moment was less than 20% of maximum value in the

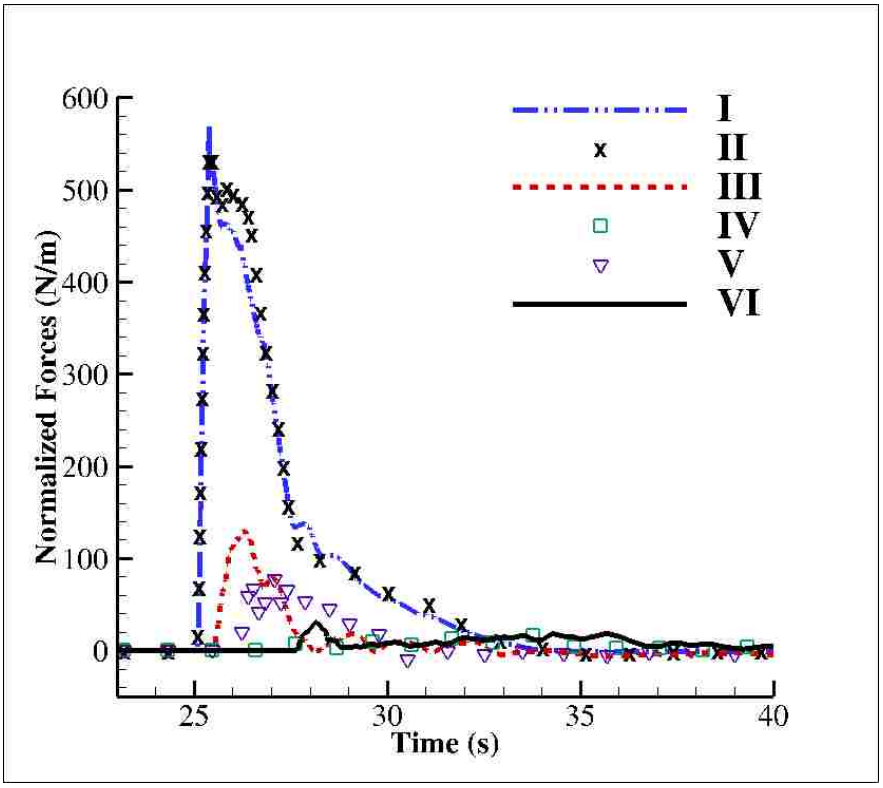


Figure 3.15: Tsunami forces on selected buildings in cross-shore direction

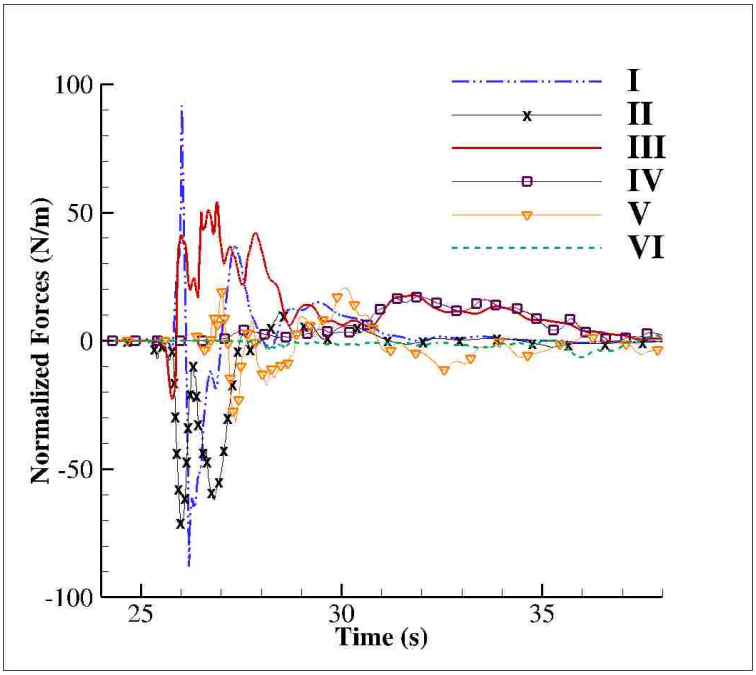


Figure 3.16: Tsunami forces on selected buildings in along-shore direction

time history of net force on the building were not plotted for clarity. Thus we see discontinuous lines. A very clear correlation between the direction of the force and the local bathymetry can be made. Buildings I and II are essentially unshielded locally, and as a result the angle of the net force remains near $0°$. Building III is shielded primarily on the south side; however, the primary force faces northward, which is somewhat counterintuitive. Figure 3.18 shows that, while water impacts the north face of Building III before the south face, the flow is not redirected along Line A, but redirected from the south face and over the top of Building I, and through interaction with the local bathymetry impacts the south wall of Building III. As the fluids flow further inland, buildings on both sides of Line A form a preferable flow channel, along which fluids are directed and flow faster than the fluids in other regions. This affects the forces exerted on building IV, V and VI in combination with shielding in front of them. For example, building IV is largely shielded by three houses, and as a result it is very susceptible to fluid forcing along its south face, resulting in an increasing net force angle as the fluid moved along Line A. Similar shielding effects were shown

for Buildings V and VI, but in a less pronounced matter as there is limited shielding directly in front of these buildings.

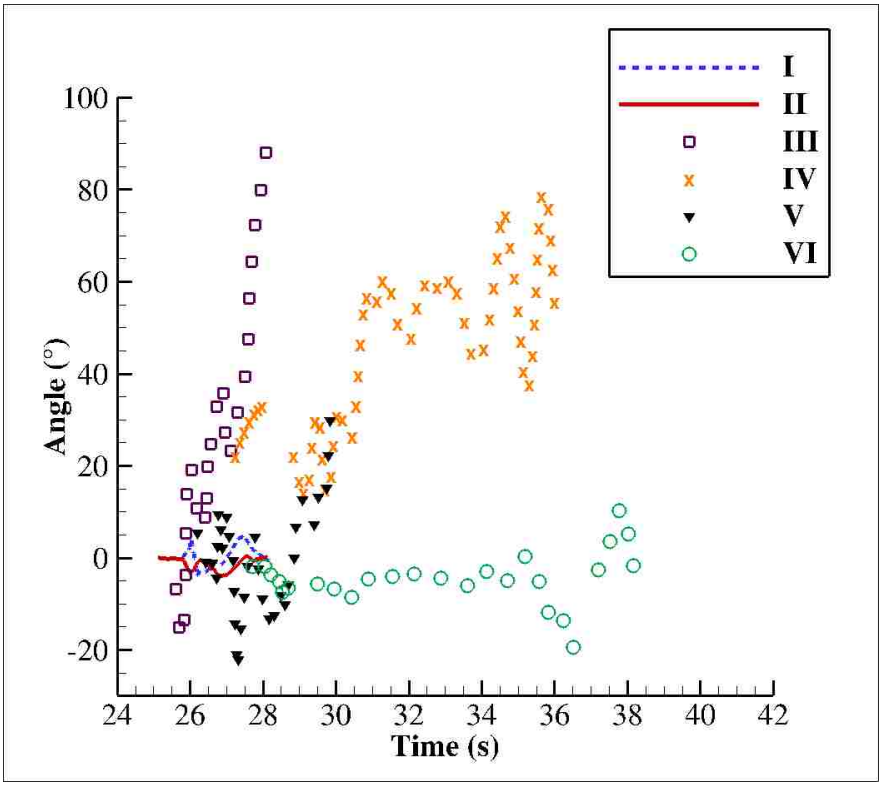


Figure 3.17: Time histories of angle of deviation of net forces on selected buildings

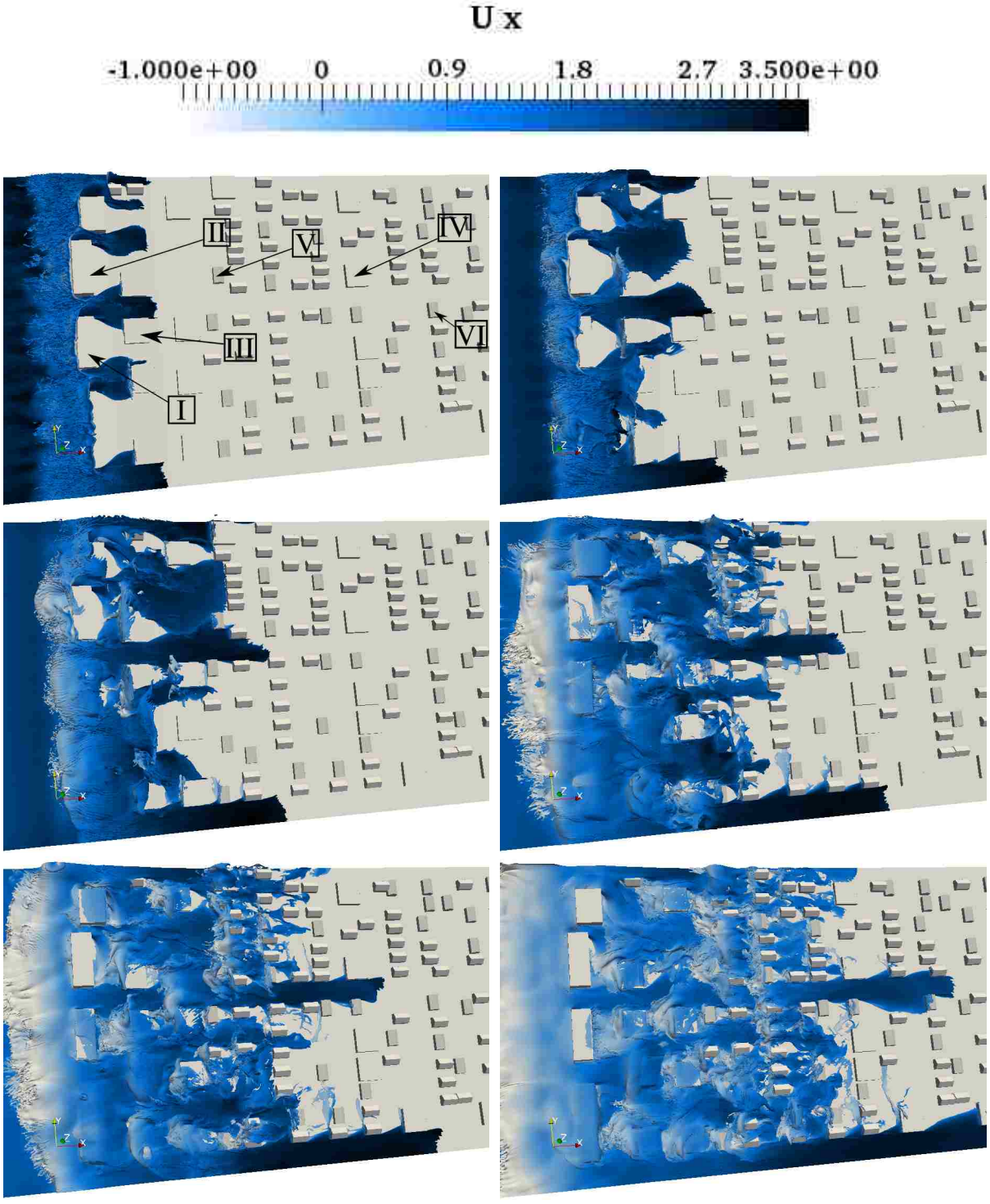


Figure 3.18: Snapshots of the simulation at 6 different moments (from left to right, top to bottom): $t = 25.4$ s, 25.7 s, 26.0 s, 26.6 s, 26.9 s, 27.5 s.

3.4 Inclusion of Constructed Environments

In some cases, prior to or immediately following construction of a new structure, it would be reasonable to evaluate tsunami impact on the structure with available data at hand from either existing experimental or numerical models which did not include the new structure. To that end, it is important that one can predict tsunami impact on the new structure from momentum flux measured at the site. To examine these scenarios, three extra cases were computed using the same computational domain that modeled subsection A but each with one selected building removed from constructed environments. In these three cases, buildings I, II and V were removed individually.

According to FEMA P-646 [Applied Technology Council, 2012], the hydrodynamic forces on a structure can be computed as

$$F_d = \frac{1}{2}\rho_s C_d A u^2 = \frac{1}{2}\rho_s C_d B(hu^2) \tag{3.20}$$

where F_d is the cross-shore component of the tsunami forces, ρ_s is the density of the fluids, A is the wet area on surface of structure normal to the direction of flow (western wall in this case), h is the water depth on the surface used to calculate wet area, u is the cross-shore component of velocity of fluids, B is the breadth of the structure in the plane normal to the direction of flow, and C_d is the drag coefficient, which may be conservatively taken as 2.0. Equation (3.20) should be used when the flow passing the structure is quasi-steady. Thus when the leading edge of the bore impacts the building, initial impulsive forces that include the inertial forces should be considered. However, the comprehensive experiments performed by Ramsden [1993] showed no significant initial impulsive force if the bed is dry before bores come due to the relatively mild-slope front profile. Even in cases where the bore has steep front and slams on the structure and thus where the impulsive force is significant, both Ramsden [1993] and Arnason [2005] observed a maximum slamming force of $\frac{3}{2}\rho_s A u^2$ in their independent and separate experiments, which is equivalent to setting $C_d = 3$ in equation (3.20). Thus by choosing a proper value for C_d, equation (3.20) can still be used to estimate tsunami forces on onshore structures [Yeh, 2006]. In this study, a drag coefficient of 2.0 was chosen because the case here is similar to the case of a dry-bed surge and it produced a resultant force that agreed well with the numerical and experimental results. For these

reasons, hereafter equation (3.20) is used to predict tsunami forces on structures from momentum flux for the entire time range and to compute the drag coefficient C_d with a formula derived from equation (3.20):

$$C_d = \frac{2F_d}{\rho A u^2} = \frac{2F_d}{\rho (hu^2) B} \tag{3.21}$$

This is consistent with traditional design code equations for hydrodynamic forces, such as those found in FEMA P-646 [Applied Technology Council, 2012]. Figure 3.19 shows a comparison of the forces in the cross-shore direction directly predicted by pressure field from the 3D numerical models and those computed with equation (3.20) from the momentum flux. The momentum flux, hu^2, in the right-hand side of equation (3.20) was calculated by measuring surface elevation h and u at the center of the footprint of the specific buildings that were removed from the domain. It is clear that by choosing $C_d = 2.0$, the agreement in the peak value between the results from the two approaches for buildings I and II were satisfactory, although forces predicted from the drag coefficient for building V were first slightly underestimated then overestimated and forces for buildings I and II were underestimated after the peak (from $t = 26$ to $t = 28$). It is also worth noting that equation (3.20) and (3.21) do not take into account the flow runup at buildings, which will influences the forces exerted on them, and assume fluids do not flow over the top of structures, which is not always true during the inundation according to the current numerical result. This can be shown in figure 3.18, where building I and V are almost completely submerged for sometime. The flows over roofs of buildings I and V carry some momentum in the cross-shore direction, which should have been exerted on them if they were high enough and did not get overtopped. Thus we should expect larger forces (by integrating pressure) on buildings like I and V if their heights increase such that they do not get overtopped.

To examine the accuracy of the assumed drag coefficient of 2.0, time histories of the drag coefficient computed directly from the numerical model using equation (3.21) are shown in figure 3.20. In the figure, it is clear that the drag coefficient for each of the three buildings oscillates around 2.0, although there are notable fluctuations between 2.0 and 4.0. Note that data in this plot only cover time range where forces are not so small compared with their peak values as not to provide numerically insignificant results that would occur due to denominators approaching zero

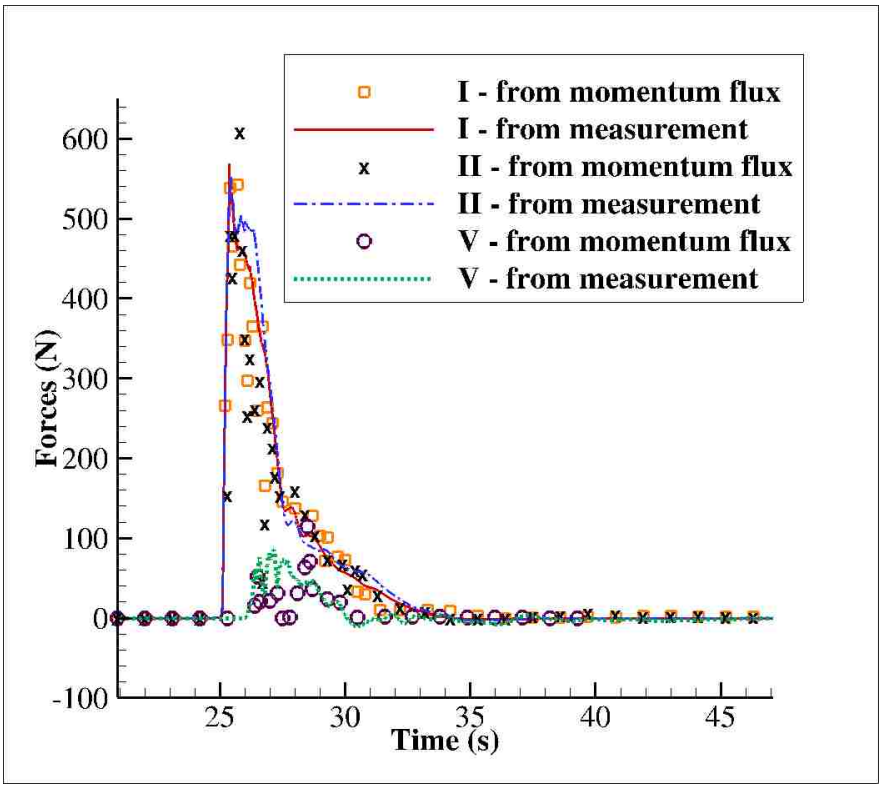


Figure 3.19: Comparison of forces in cross-shore direction. (from momentum flux: forces computed from momentum flux measured in the case without constructed environments; from measurement: forces computed by integrating pressure on surface of objects in numerical model)

or negative drag coefficients resulting from offshore flows.

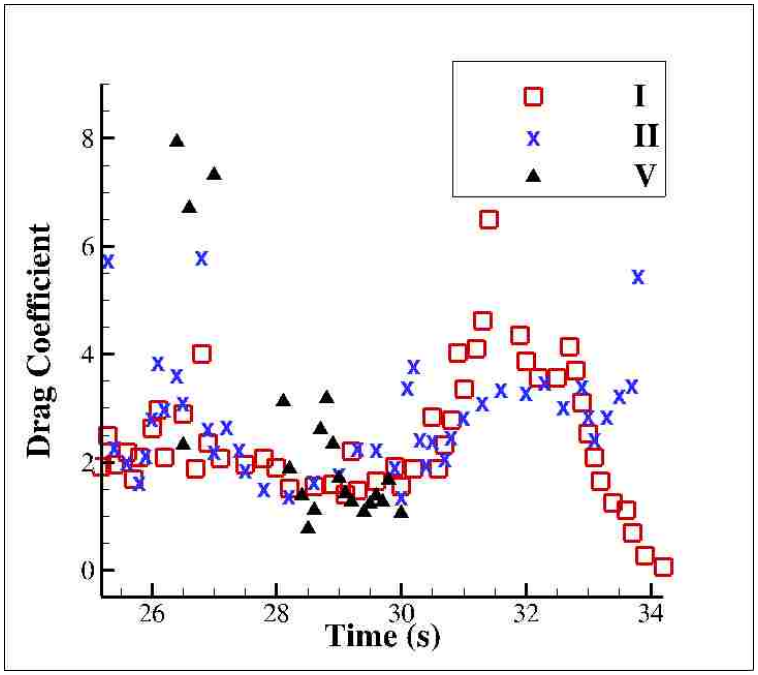


Figure 3.20: Time histories of the drag coefficient

These results show that predicting tsunami loads on planned or newly built structures from momentum flux based on old inundation data (and with the recommended values for the drag coefficient by FEMA P-646 [Applied Technology Council, 2012]) is acceptable, and the addition of the new structure to the constructed environment would not be expected to change the local inundation values substantially.

However, before the analysis described above can be performed, local inundation data is necessary. One of the primary research obstacles in predicting tsunami forces on structures is that tsunami inundation models often do not include aspects of the constructed environment. A numerical case was developed in which all buildings were removed from the model to investigate the influence of incorporating constructed environments into modeling of tsunami inundation. Surface elevation and velocity were measured at center of the footprint of Buildings I-VI and corresponding momentum flux was also computed.

Using the same approach to generate forces as described above for figure 3.19, forces were predicted for buildings I-VI based on results from this "bare-earth" case. Figure 3.21 shows that forces predictions at building I and II agree well with the peak value measurements. However, for other selected buildings, forces get overestimated at almost all times. This can also be reflected as shown in figure 3.22 if we plot time history of drag coefficient computed from forces and momentum flux in this bare-earth case with the same approach as in figure 3.20 (F_d is the force from integrating predicted pressure on the surface of buildings in the previous cases where all buildings were included, while hu^2 is the momentum flux from the bare-earth case where all buildings were removed). Before $t = 30$ s, drag coefficients for buildings I and II oscillate around 2.0, while for all other buildings, drag coefficients are as low as 1.0, which indicated that if tsunami inundation is modeled without constructed environments and that inundation data is used to predict tsunami loads on buildings, these loads can be greatly overestimated. This is not the case for buildings closest to shoreline, where the effects of constructed environments are largely negligible during the initial stages of impact. In the bare-earth case, water depth and velocity in the region behind buildings I and II are higher than the predicted values in cases with constructed environments incorporated, since these buildings along the shore create large levels of resistance to inundation of the tsunami.

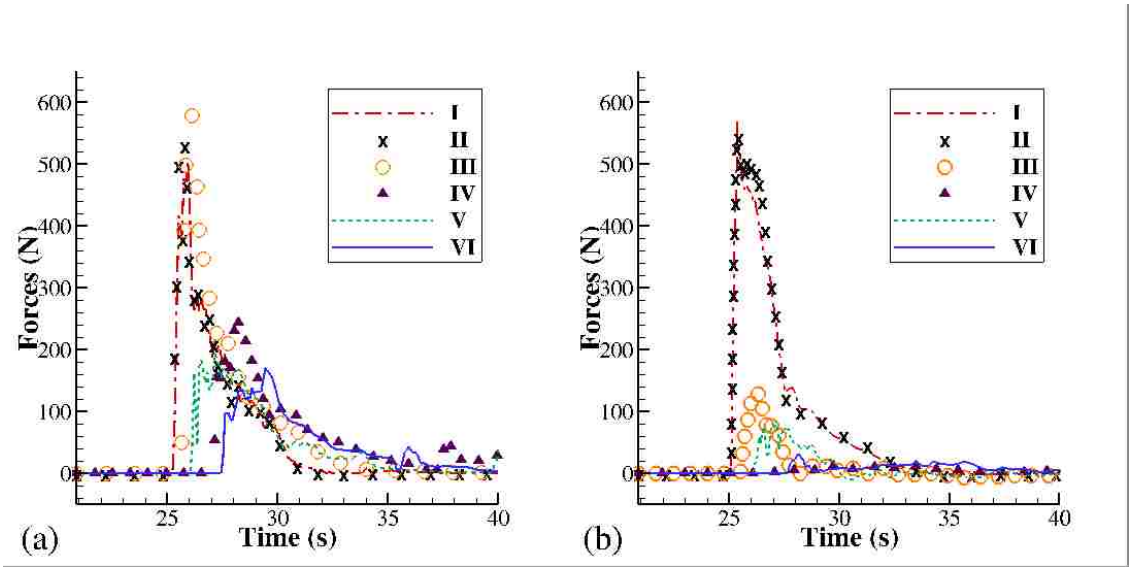


Figure 3.21: Comparison of forces in cross-shore direction: (a) forces computed from momentum flux measured in the case without constructed environments; (b) forces computed by integrating pressure on surface of objects in numerical model (with constructed environments)

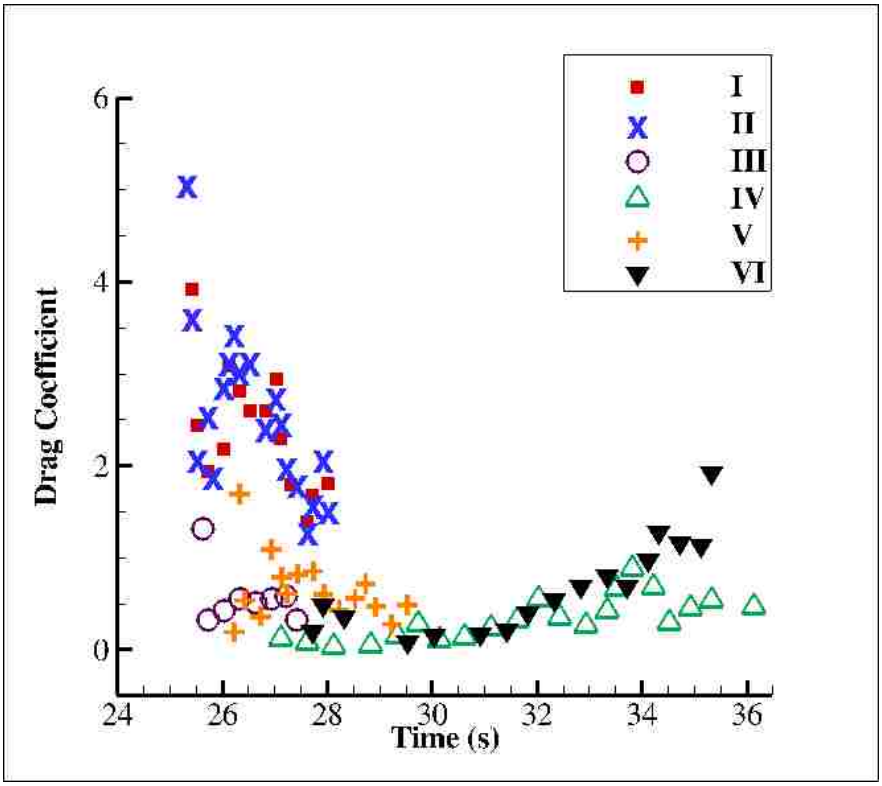


Figure 3.22: Time histories of the drag coefficient in bare-earth case

3.5 Localized Building Forces

One final phenomena that must be considered is the local effect of wave impacts on individual structural components. While the net force on the building provides valuable information about system loss, forces on individual walls can show localized forces that cancel one another out when considering the total force on a building and yet can cause significant damage. Figure 3.23 shows the along-shore force time histories on each individual face of Building II and the corresponding total force time history. It is evident that equal and opposite forces on the north and south walls are 2-3 times larger than the net force on the system, suggesting a susceptibility to local damage.

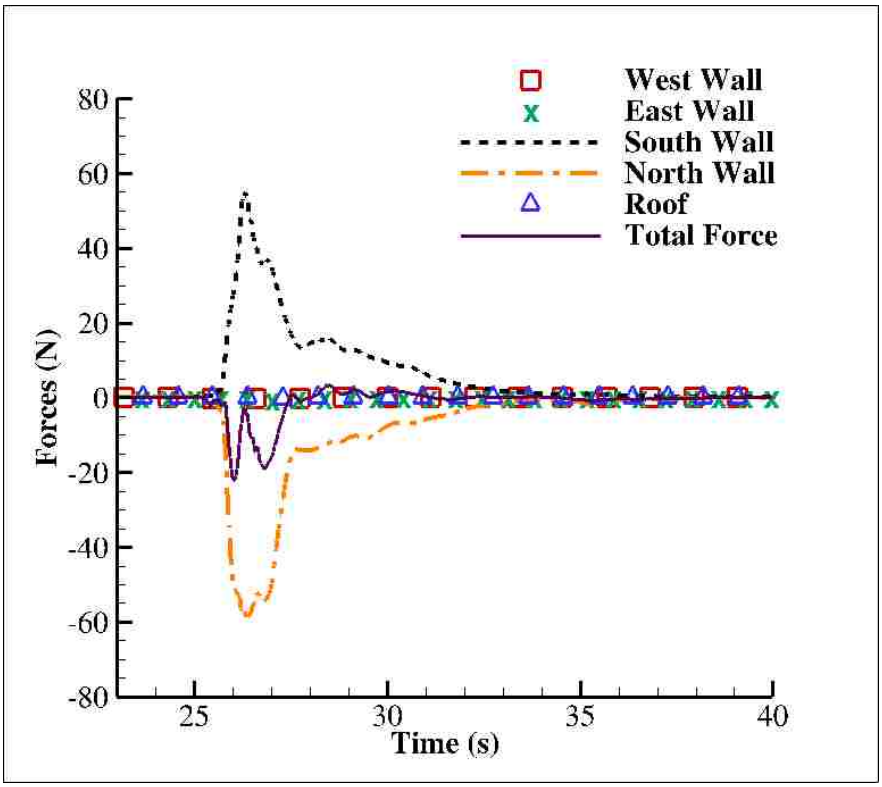


Figure 3.23: Tsunami forces on individual walls of building II

3.6 Conclusions

In this chapter, a 3D numerical model of tsunami inundation within a model-scale constructed environment was developed using a 3D RANS approach. Free surface elevation, velocity, and momentum flux were measured and compared with experimental data. The prediction of these quantities from the numerical model agrees well with experimental measurements, except for peak value of velocity and momentum flux, the discrepancies in which were discussed in detail. Tsunami forces on several representative structures were also computed by either integrating pressure and shear stress on their walls or extrapolating from flow parameters. The primary conclusions of this chapter are:

1. The entire basin can be modeled using subsections with proper width without loss of accuracy in areas of interest. Increasing the width of these subsections does not necessarily increase quality of the predicted results in many cases (figure 3.11), which can be critical when extrapolating these ideas to "real-world" scenarios where modeling of an entire town can be computationally impractical.

2. The optical method used in the experiment to obtain the peak velocities can lead to misleading force predictions, as the velocity field data from the simulation revealed that the assumption that the velocity of the leading edge of the bore corresponds to the maximum flow speed is not necessarily valid. This is quite important, as surveys following recent tsunami events often relied upon video forensics to predict the flow speeds that could be used to extrapolate force predictions.

3. The amplitude and direction of local forces on structures showed a high correlation with local constructed environments. Selected buildings revealed quite large force components even in the along-shore direction, which may cause local failure of the structure during a tsunami.

4. The traditional approach to predicting forces on structures via definition of a force coefficient

using a recommended value of 2.0 for the drag coefficient can be problematic when the constructed environment is not considered. Results herein showed that such a recommended coefficient is not practical since the drag coefficient not only varies with time but also highly depends on location and the surrounding bathymetry which can control the direction of the flow.

5. With influence of the constructed environment, different walls of individual buildings can be impacted by the flow at different times. Thus even when the net force is not very large, forces on individual walls and components can be quite large and result in critical local damage.

Ultimately, this chapter presents an important step in recognizing the significance of considering the effects of constructed environments on the prediction of tsunami forces within an inundated area. While this proof-of-concept study does not include the intricacies of an actual constructed environment, these results are nonetheless important in demonstrating these effects.

Chapter 4

THE TWO-DIMENSIONAL SEASIDE MODEL

In this chapter, a two-dimensional (2D) model for the Seaside problem described in chapter 3 is developed and compared against the three-dimensional (3D) Seaside model developed in that chapter. Additional results from the 3D Seaside model are shown for comparison and discussion. The 2D model is based on the nonlinear shallow water equations (NSWE), which is solved with Adaptive Mesh Refinement (AMR) by using the GeoClaw software [Berger et al., 2011, Clawpack Development Team, LeVeque et al., 2011].

A simpler dam-break experiment described in section 3.1 is first simulated. Then the 2D model is used to simulate the same wave tank experiment detailed in chapter 3. In section 4.3.1, details of the setup for the 2D model is presented. In section 4.3.2, the generated numerical waves from the 2D model is shown and compared against the 3D model. Flow parameters at representative wave gauges in different regions of the domain are then presented and discussed, followed by prediction of tsunami loads on selected buildings during the tsunami inundation. The chapter ends with some conclusions in section 4.4.

4.1 Methodology

The nonlinear shallow water equations have been used broadly by many researchers in modeling of tsunamis, storm surge, and flooding. It can be written in the form of a nonlinear system of

hyperbolic conservation laws for water depth and momentum:

$$h_t + (hu)_x + (hv)_y = 0, \tag{4.1a}$$

$$(hu)_t + \left(hu^2 + \frac{1}{2}gh^2\right)_x + (huv)_y = -ghB_x - Dhu, \tag{4.1b}$$

$$(hv)_t + (huv)_x + \left(hv^2 + \frac{1}{2}gh^2\right)_y = -ghB_y - Dhv, \tag{4.1c}$$

where $u(x, y, t)$ and $v(x, y, t)$ are the depth-averaged velocities in the two horizontal directions, $B(x, y, t)$ is the bathymetry/topography, and $D = D(h, u, v)$ is the drag coefficient. The subscript t represents a time derivative, while the subscripts x and y represent spatial derivatives in the two horizontal directions. The value of $B(x, y, t)$ is positive for topography above sea level and negative for bathymetry. Coriolis terms can also be added to the momentum equations but is generally negligible for tsunami problems and is not used here. The drag coefficient used in the current model is

$$D(h, u, v) = n^2 g h^{-7/3} \sqrt{u^2 + v^2}, \tag{4.2}$$

where n is the *Manning coefficient* and depends on the roughness of the ground. A constant value of $n = 0.025$ is often used for tsunami modeling, and this value is used for all problems in this dissertation. The above equations are written in Cartesian coordinates and rectangular grids in these coordinates can be used for modeling tsunamis in small regions. Logically rectangular grids mapped to spherical coordinates can be used to model transoceanic tsunami propagation.

The nonlinear shallow water equations are solved by using the GeoClaw software [LeVeque et al., 2011, Berger et al., 2011] in this dissertation, which features adaptive mesh refinement (AMR) and is released as a submodule of the Clawpack software [Clawpack Development Team], an open source package for solving hyperbolic systems of partial differential equations (PDEs) of one, two and three dimensions, through finite volume implementation of high-resolution Godunov-type "wave-propagation algorithms". Cell averages of the solution variables q are computed over the volume of each cell and updated with waves propagating into the cell from all surrounding cell edges. The wave at each edge is computed by solving a "Riemann problem" with initial piecewise

constant data determined by cell averages on each side of the edge. This method is especially good at solving problems with discontinuous solutions like shock waves, which usually arise in the solution of nonlinear hyperbolic equations (e.g. bores in the case of NSWE).

Specifically, GeoClaw uses a variant of the f-wave formulation of the "wave-propagation algorithm" that allows incorporation of the topography source terms on the right hand side of equations (4.1b) and (4.1c) into the Riemann problem directly. The augmented Riemann solver in GeoClaw combines the desirable qualities of the Roe solver [Roe, 1981], HLLE-type (Harten, Lax, van Leer and Einfeldt) solvers [Einfeldt, 1988, Einfeldt et al., 1991] and the f-wave approach [Bale et al., 2003]. The Roe solver provides an exact solution for the single-shock Riemann problem. It is also depth positive semidefinite like the HLLE solves, has a natural entropy-fix by providing more than two waves and yields a better approximation for problems with large rarefactions. A large class of steady states is also preserved, even for non-stationary steady states with non-zero fluid velocity. In addition, it is able to handle the presence of dry states in the "Riemann problem", in which one state is wet ($h > 0$) while another is dry ($h = 0$), or both states are dry. It also works robustly in situations where the topography changes abruptly from one cell to another by an arbitrarily large value. For more details of the augmented Riemann solver in GeoClaw, see George [2008].

A typical characteristic of tsunami inundation models, especially those that incorporate constructed environments, is that the spatial scale of regions of interest may vary from kilometers to meters. For regions in the open ocean, grid cells can be tens of kilometers on a side, while for regions near the shoreline or in constructed environments onshore, grid cells must be refined to several meters or less, since the size of a building may be only several meters and an adequate number of grid cells are required to achieve acceptable accuracy. In GeoClaw, a patch-based AMR technique can efficiently handle these situations [LeVeque et al., 2011, Berger and LeVeque, 1998]. In regions where finer grids are needed, patches of level 2 (probably with smaller area) overlap base-level patches to provide finer grids. These patches can be overlapped further by patches of higher levels until a sufficiently fine resolution is reached. During the computation, grids are refined in specific regions automatically as time evolves. This is done by flagging cells where higher resolution is needed. These flagged cells are then clustered into refinement patches. The criteria for

flagging in the current model is to flag cells where the surface elevation is perturbed from sea level beyond a specified threshold. The tolerance for refinement, minimum and maximum allowable level of refinement can be specified before the computation to control the refinement during the simulation. In addition to this, cells in some user-specified rectangular regions can also be refined to a required level regardless of the criteria of refinement mentioned above.

The "wave-propagation algorithm" in GeoClaw is explicit and hence a CFL number of less than 1 is required for stability, which essentially limits the length of the time step. The time step on base-level patch is chosen to satisfy the CFL condition while on patches of higher levels, the time step is refined from the base-level time step by the same factor as in space. For example, if level 2 grids are refined in both the x and y direction by a factor of 2 relative to level 1, then the time steps on all level 2 grids must be half of the time steps on level 1 grids. GeoClaw is designed to first advance one time step on level 1 grids, and then update the solution on level 2 grids by several time steps (2 for the example above) and so forth for grids of higher levels.

A more detailed description of the algorithms in GeoClaw and AMR is presented in chapter 5.

4.2 A Simple Dam-break Problem with Bore-Structure Interaction

A simple 2D model was first built to model the experiment introduced in section 3.1. The 2D model was built with the GeoClaw software described in section 4.1. In the 2D model, the column was incorporated into the computational domain through the topography term $B(x, y)$ on right hand side of equations (4.1b) and (4.1c). Values for $B(x, y)$ are set to a very large constant value, h_c, in the region of the column and to 0 elsewhere. This prevents water from overtopping the area, thus simulating a column. Setting h_c to a very large value also made all four side walls of the square column be more "vertical" in the model since they are represented by steep slopes arising from $B = 0$ (outside the column) to $B = h_c$ (inside the column). The coarsest level grid had a resolution of 0.02 m by 0.02 m and covered most of the computational domain; the finest mesh near the column was 0.25 cm by 0.25 cm.

First, a case without the column was modeled. Figure 4.1 shows time histories of the water level at 5.2 m downstream from the gate (i.e., at $x = 11.1$, the center of the column) from the 2D

GeoClaw model, the 3D OpenFOAM model built in chapter 3 and the experiment. In general, both 2D and 3D models accurately predict the arrival time of the bore, which is at $t = 3.2$ s.

The 3D OpenFOAM model matches the measurement better than GeoClaw with a sharp (but not vertical) slope at the front, a gradually rising surface to the peak near $t = 8$ s, then a downward slope, followed by interactions with the reflected wave from the back wall that creates the second jump in the water level at around t = 14 s.

The 3D OpenFOAM model models water viscosity, which diffuses sharp discontinuities. In contrast, the 2D GeoClaw model does not model that and solutions of the nonlinear shallow water equations for the dam-break problem with an initial discontinuity yields a shock wave (discontinuity) propagating to the right as a vertical bore front followed by a region with constant water depth; as a consequence, GeoClaw slightly overestimates the initial height of the bore front, underestimates the height at t= 8 s, and presents the reflected wave as a second sharp discontinuity at t = 13.1 s.

At the same location, streamwise (the along-channel direction) components of the velocity at different depths were also predicted. Figure 4.2 shows time histories of streamwise velocity at 9 different distances from the bottom. Note that since the 2D model is depth-averaged, its predicted velocity is constant with depth. Near the water surface, the prediction from the 2D model matches the measurements very well except for the spike at the front, which is captured by the 3D model.

Figure 4.3 shows a comparison of total forces on the square column from the experiment, the 3D model and the 2D model. The force predicted by the 3D model was obtained by integrating the pressure and viscous fluid forces on the surface of the column (See equation (3.13)). The 3D model predicts the force very well in terms of magnitude and is able to capture even the small spike near $t = 4$ s. In the 2D model, no pressure field is computed and available for force prediction. To predict forces from the 2D model, data from the previous case without the column was used instead. The water level, h, and streamwise velocity, u, were first sampled at the center of the footprint of the column that was removed from the domain, to compute the momentum flux, $M = hu^2$. Then forces were computed from equation (3.20) with the same choice of $C_d = 2.0$. Note that in the experiment or 3D model, the water level on the upstream side of the column is different from that

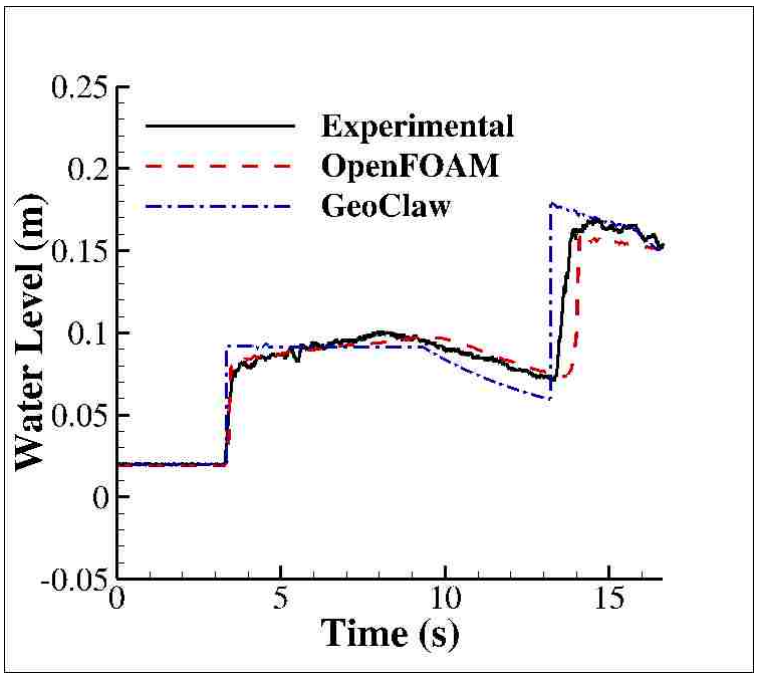


Figure 4.1: Time history of the water level at 5.2 m from the gate (center of the column) with the column removed

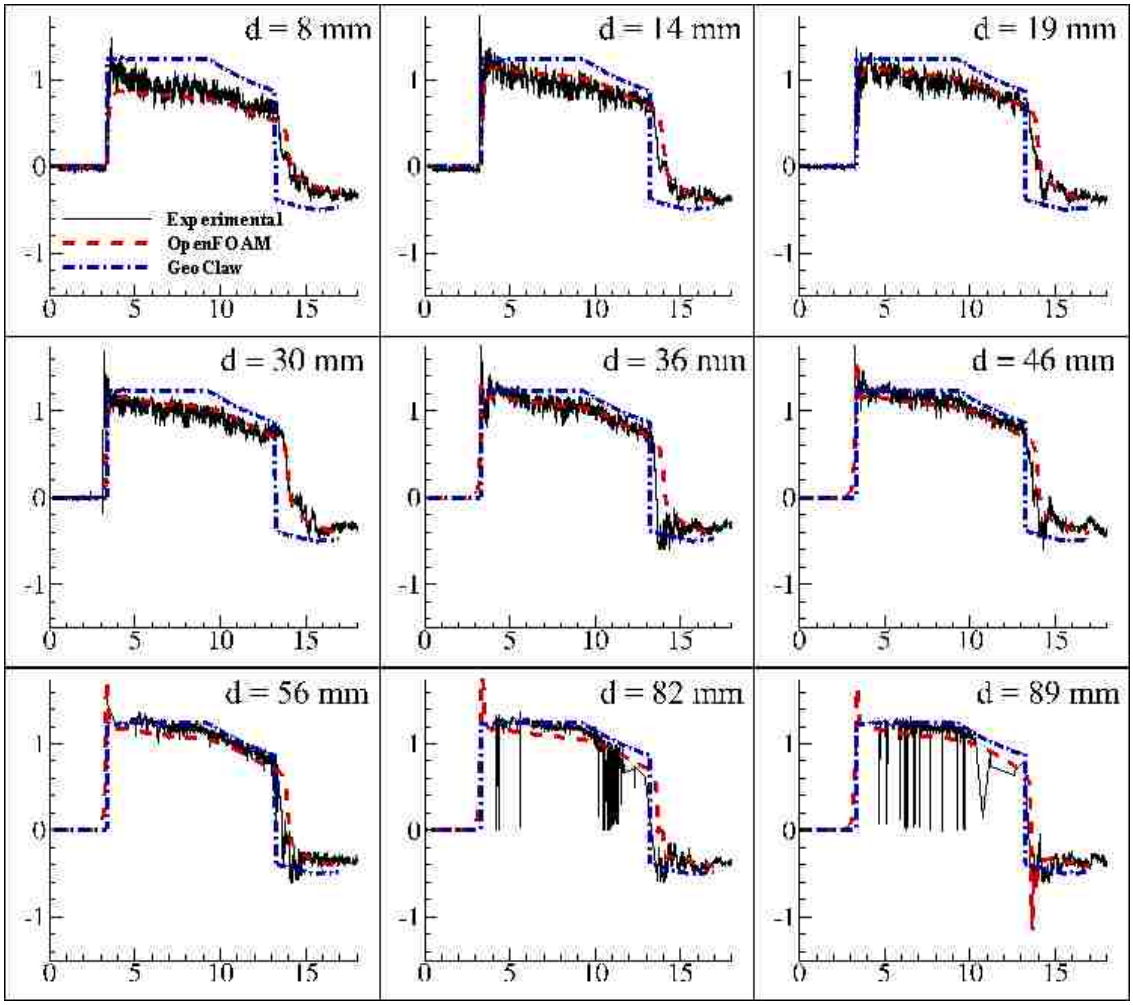


Figure 4.2: Time history of streamwise velocity at different distances, d, from the bottom at 5.2 m from the gate (center of the column) with the column removed. Abscissa: time (s). Ordinate: velocity (m/s).

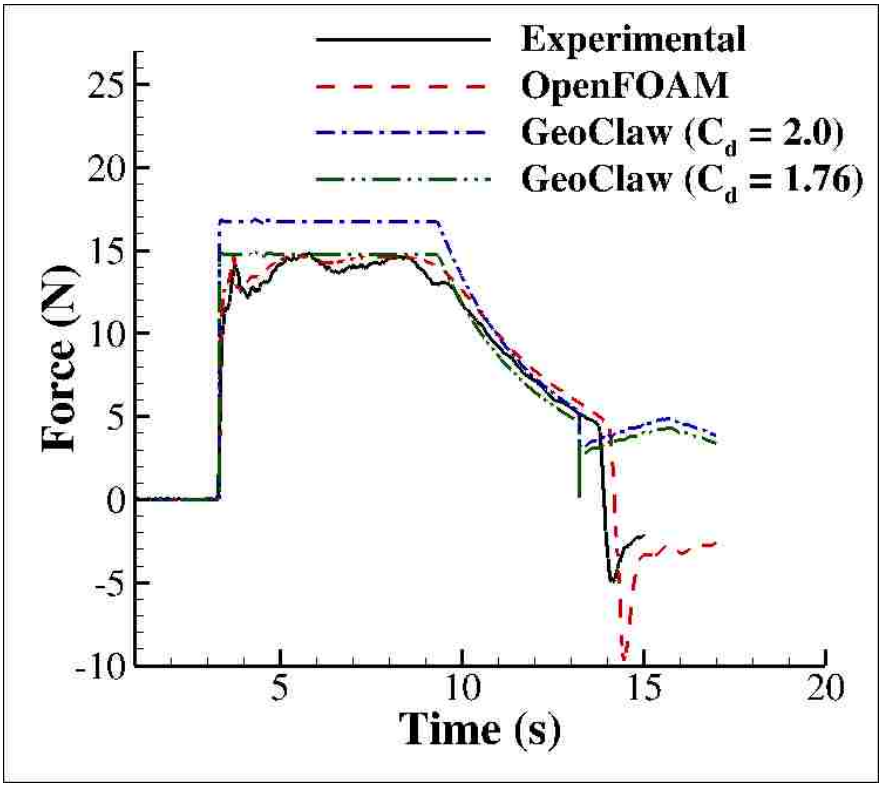


Figure 4.3: Comparison of measured and predicted horizontal forces on the square column

on the downstream side of the column. This causes a difference in hydrostatic pressure and thus a hydrostatic force on the column. For this reason, it may be more appropriate to refer to this value as the coefficient of resistance instead of solely as a drag coefficient. Using a drag coefficient of 2.0 overestimates the force by 13% in general. This is as expected since it is said to be "conservative" according to FEMA P646 [Applied Technology Council, 2012]. Figure 4.3 also shows that if a drag coefficient of 1.76 is used instead, the force prediction from the 2D model matches the measurement more closely.

4.3 The Seaside Model

A 2D inundation model is developed for the Seaside problem described in chapter 3, based on the GeoClaw code. This section describes the setups of the 2D model and quantities of interest output by the 2D model. The output are compared to those generated by the 3D model developed in chapter 3.

4.3.1 Model Setups

To solve the NSWE, the GeoClaw code Berger et al. [2011], LeVeque et al. [2011] is used. Unlike the 3D OpenFOAM model, the 2D GeoClaw model can model the entire basin due to relatively cheaper computational cost of the governing equations that are solved. Thus, the computational domain is a 48.8 m by 26.5 m rectangle. The geometry of the basin bottom and constructed environments are described by topography files, which specify $B(x, y)$ on the right hand side of equations (4.1b) and (4.1c). Typical computational time for one simulation is approximately six hours with a single core in an Intel(R) Core(TM) i7-4790 CPU processor. Note that the computational resources required by the GeoClaw model is only $\frac{1}{2500}$ of what is required by the 3D OpenFOAM model in this study.

To generate tsunami waves in GeoClaw, user defined time varying boundary conditions can be specified at the inlet of the computational domain, based on data for the wavemaker speed $s(t)$ in the physical experiment. The data from the physical experiment can be fit quite well with a

Gaussian of the form

$$s(t) = Ae^{\beta(t-t_0)^2} \tag{4.3}$$

with $\beta = 0.25$, $t_0 = 14.75$ and amplitude $A = 0.51$. However, several trials resulted in a better match at wave gauges WG1, WG2, WG3, and WG4 by setting $A = 0.6$, which was therefore used for all simulations.

The adaptive mesh refinement (AMR) feature of GeoClaw was used, with a mesh size for the base-level grid of 0.5 m (corresponding to 25 m in full scale) in both cross-shore direction and along-shore direction. The term cross-shore is used to refer to the direction that the wave propagates from the wavemaker to the structures onshore, while the direction perpendicular to the cross-shore direction is referred to as the along-shore direction. The mesh is refined in the nearshore region up to 4 levels, with specified refine ratios: 4 for from level 1 to 2, 5 for from level 2 to 3 and 2 for from level 3 to 4. The finest mesh in the domain with this setup for AMR is 0.0125 m by 0.0125 m (corresponding to 0.625 m in full scale) and eventually covers the entire onshore region.

4.3.2 Comparison of Offshore Flow Parameters

Figure 4.4 shows the time history of the free surface elevation at two wave gauges offshore from the 2D model, plotted against the result from 3D model and experimental measurement. Velocity of the piston is derived by taking the time derivative of the displacement history and is used to specify input velocity at the inlet boundary of the 2D model for wave generation. Water level agreement between the measured and modeled elevation was satisfactory, overall. Although both models slightly underestimate wave height at gauge WG3 and propagation speed of wave, based on the scatter and uncertainties in the experimental results and the qualitative agreement between the models and the experimental data, the numerical wave considered in the models is sufficient for this work.

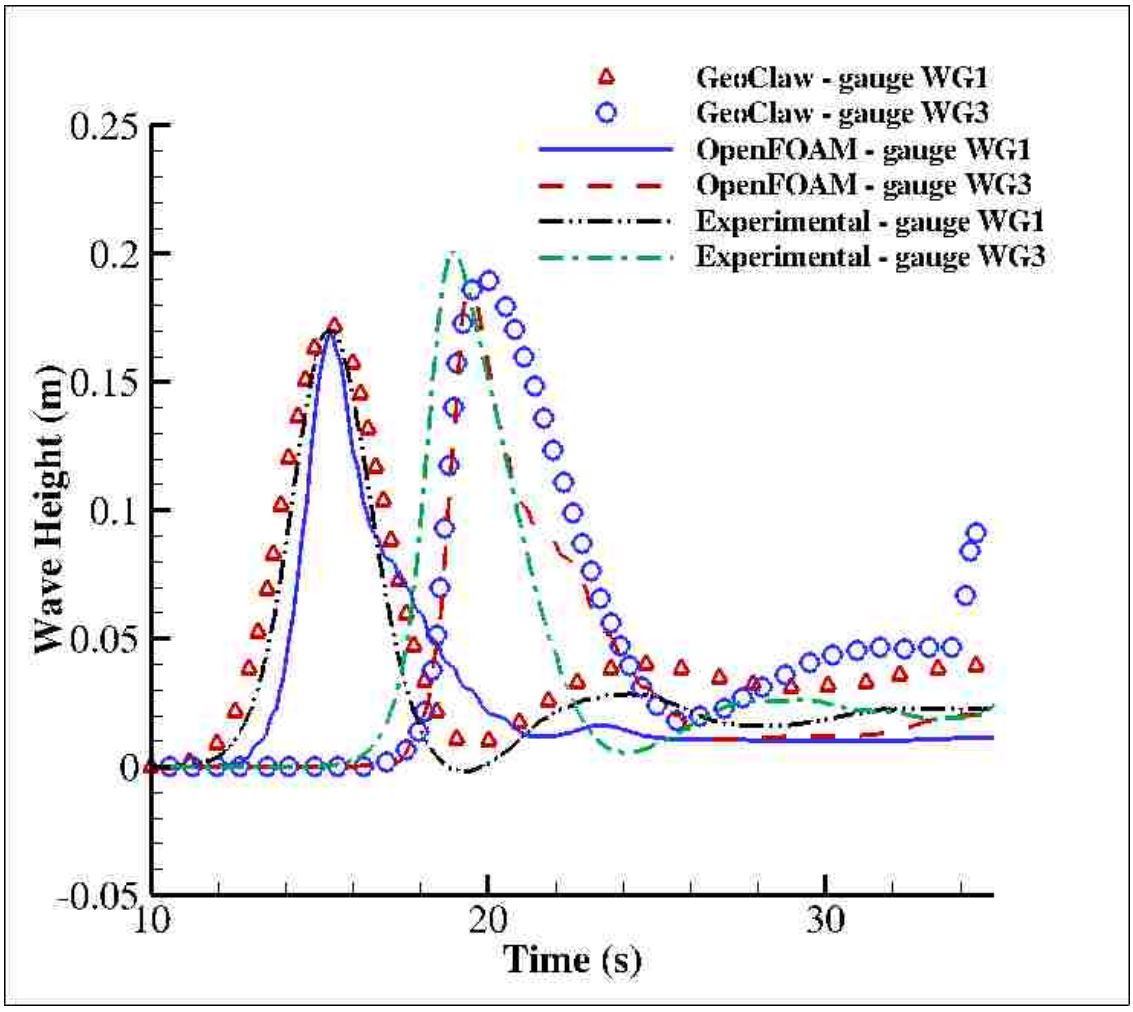


Figure 4.4: Time histories of surface elevation at gauge WG1 and WG3 from the 2D model. Data from figure 3.10 is re-plotted here for comparison.

4.3.3 Comparison of Onshore Flow Parameters

Onshore time histories of the free surface elevation, cross-shore velocity and corresponding momentum flux at selected on-shore gauges are shown in Figs. 4.5-4.8. Note that the 2D model models the entire basin and produces flow parameters at all gauges in one simulation, while the 3D model models one of the four subsections denoted in figure 3.7 and produces flow parameters at one of the four groups of gauges in each simulation. The width of the four subsections are chosen to be similar to the width of the wider subsection in chapter 3 to minimize the effect of boundaries. After the peak (initial impact), there appears to be a significant drop in discrepancies between modeled and measured water level and fluid velocity; therefore, the discussion that follows will separately compare the results before and after the peak.

Onshore Time Series Near Initial Impact

The water level amplitude by OpenFOAM and arrival time by both OpenFOAM and GeoClaw agree fairly well with measurements at many of the gauges in groups A, B and C, but GeoClaw underestimates the amplitude at many gauges. These differences reflect the challenge of modeling a turbulent and rapidly varying bore front. An additional factor is that the gauges in groups A, B and C are placed along straight lines, representing roads within the community, whereas those in group D are set behind buildings. As a consequence, flow around group A, B and C gauges is dominated by flow in the cross-shore direction, while flow around group D gauges is more complex and challenging to model.

Fluid velocity experimental values derived by optical means are significantly lower than the modeled OpenFOAM and GeoClaw velocity in many of the 16 cases presented in Figs. 4.5-4.8. This was explained and discussed in section 3.2.3. Here the same approach is applied to analyze the animation of GeoClaw numerical results to obtain estimates of 1.3m/s for peak velocity: figure 4.9 showed modeled velocity distributions in the bore at two consecutive time steps in the GeoClaw simulation at gauge A4, illustrating that the modeled maximum fluid occurs at some point behind the bore front.

Momentum flux modeled by OpenFOAM and GeoClaw do not agree well with experimental estimates, due to the discrepancies in fluid velocity estimates, discussed above. This is critical, since momentum flux is often used to compute the tsunami forces on structure, as discussed in detail later in this chapter.

In summary, predictions near the initial impact are challenging for both models, but the 3D OpenFOAM model performs better than the 2D GeoClaw model because it models turbulence and the variation of velocity with depth.

Onshore Time Series in Post-impact Region

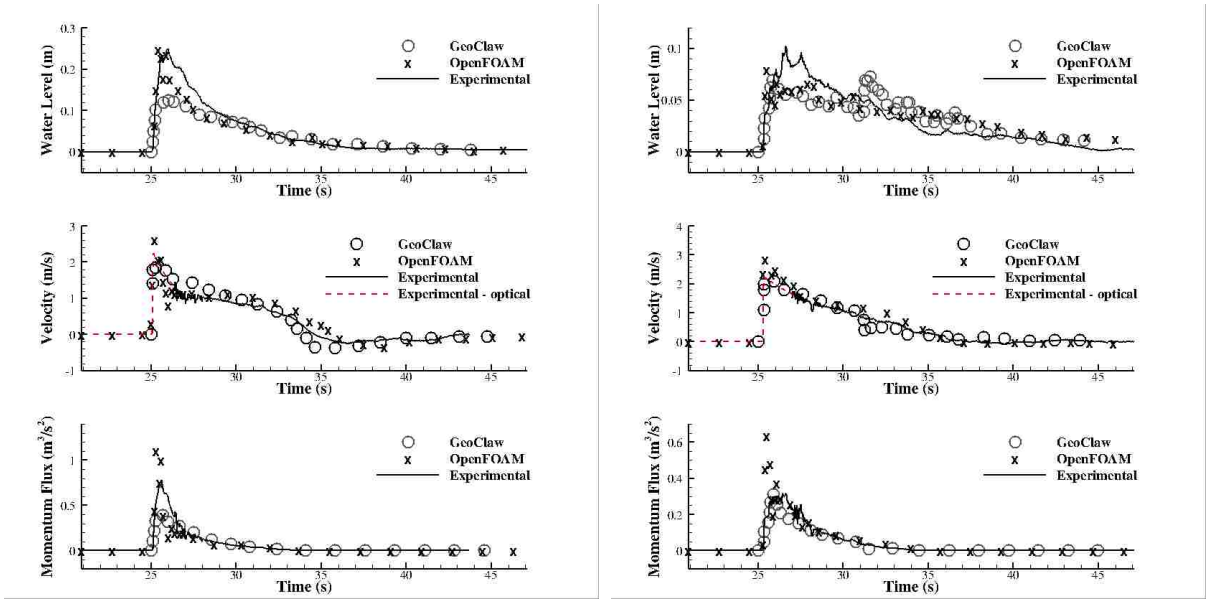
Water level agreement among both models and the experimental data are significantly improved after initial impact. Note that some gauges are quite far from the shoreline (for example, gauges A6, B8, C8), where the inundation depth is very shallow compared to the peak value near the shoreline (less than 20% of the peak value). Even at these locations, however, both numerical models provide reasonable predictions. It is also of interest that, as noted above, GeoClaw predicts a lower bore front propagation speed than OpenFOAM; as a result, arrival of the OpenFOAM bore front agrees well with experiment, but the GeoClaw bore front is significantly delayed at gauges farther inland, such as B8 and C8 (Figs 4.6d and 4.7d). This is also consistent with the slower propagation speed of the offshore GeoClaw wave, noted above.

Fluid velocity measurements by the ADV are more stable after 30 s, and both OpenFOAM and GeoClaw velocity time series agree much better with the experimental data at gauges in groups A, B and C. Agreement does degrade significantly in group D, especially in the case of GeoClaw; this is no doubt due to the more complicated fluid flow in the group D environment, behind buildings, compared to the relatively simpler cross-shore flow in the street environments of groups A, B and C (figure 3.7).

Momentum flux from both numerical models are in better agreement with the measurements at most gauges, since water level and velocity agreements are better than in the $t < 30s$ time period.

Figure 4.10 compares snapshots of the simulation near line A from the two models at 3 different times. The 3D model provides substantial detail about the complex flow among buildings,

including the strong channeling effect along line A, aligned with the street, and among the buildings on both sides of the street. These channeling effects can alter the forces exerted on both sides of that street, so that any differences between OpenFOAM and GeoClaw in modeling such effects may result in different prediction of forces on the buildings.



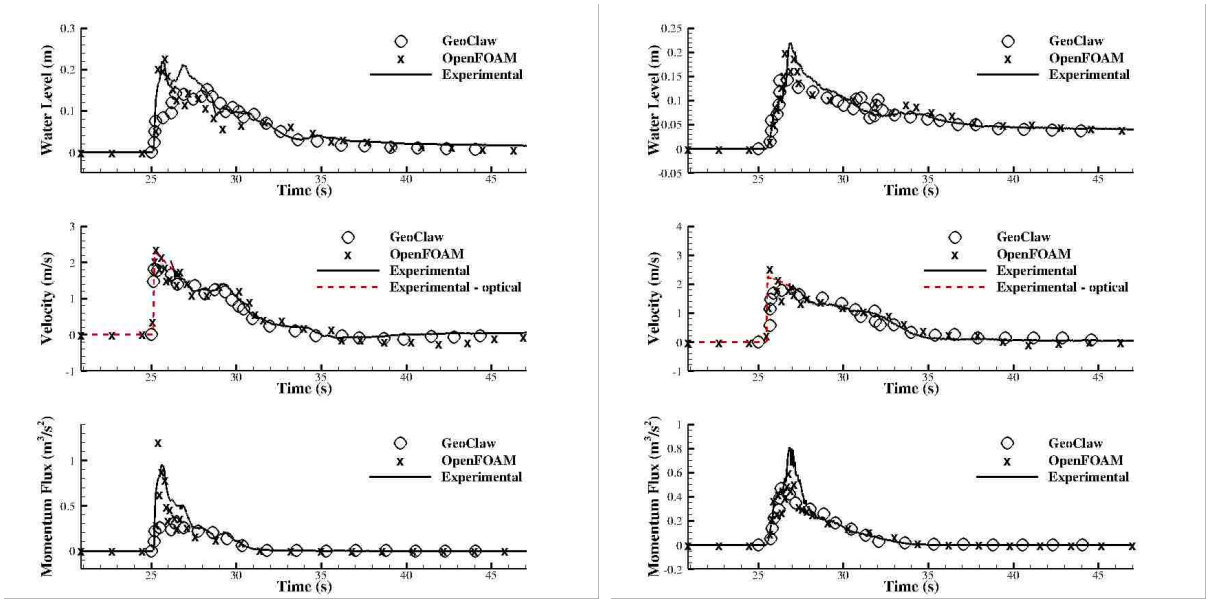
(a) Gauge A1

(b) Gauge A2

(c) Gauge A4

(d) Gauge A6

Figure 4.5: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges along line A (Note that ranges of Y axis are different in different subplots)



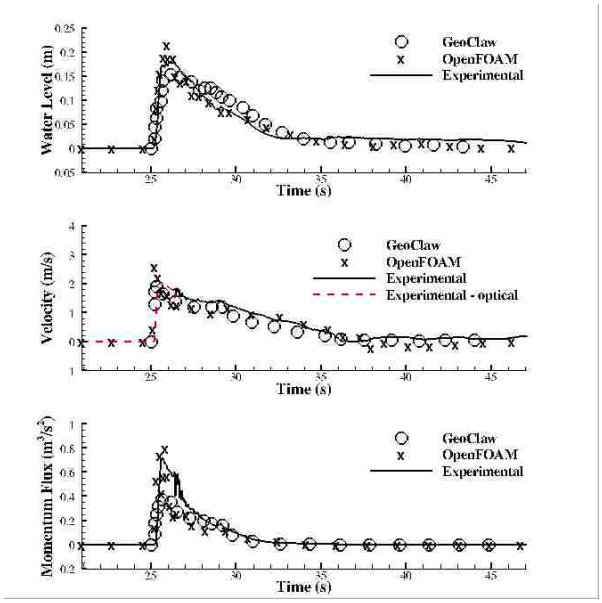
(a) Gauge B1

(b) Gauge B3

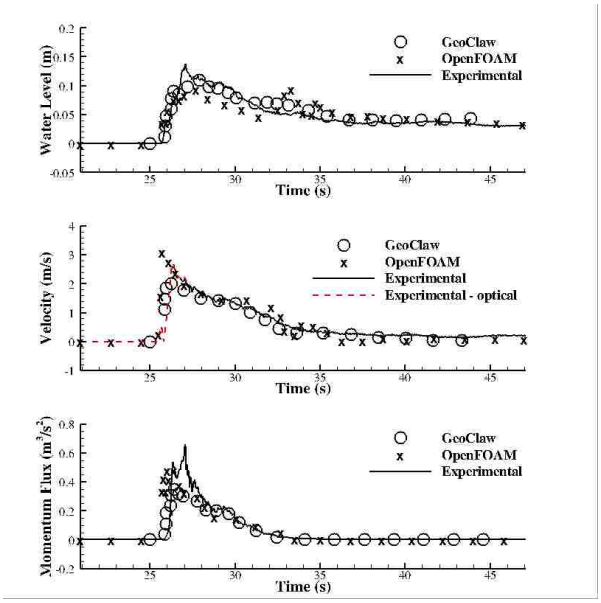
(c) Gauge B6

(d) Gauge B8

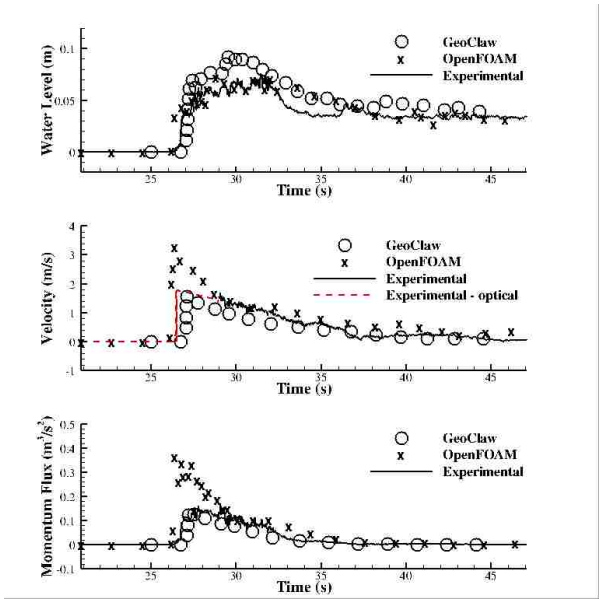
Figure 4.6: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges along line B



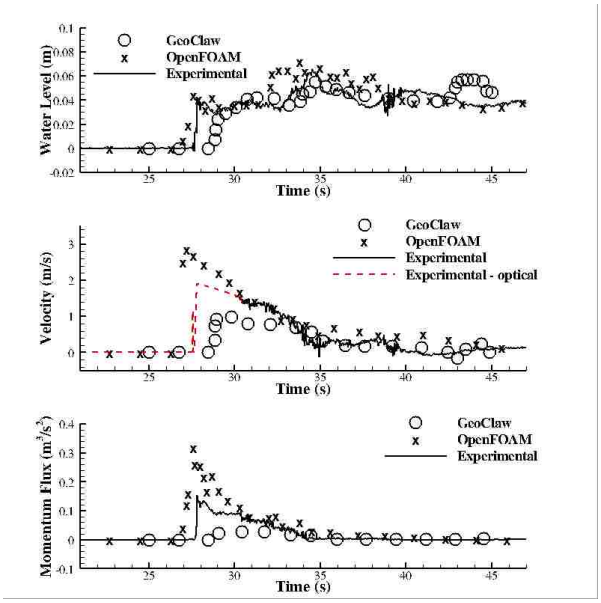
(a) Gauge C1



(b) Gauge C3

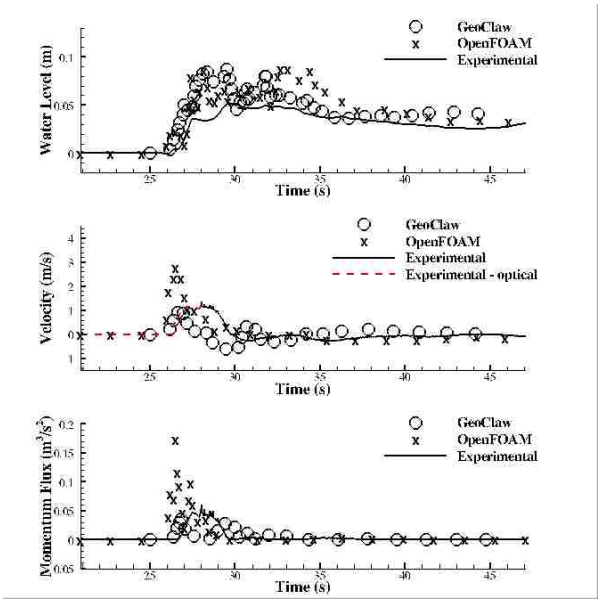


(c) Gauge C6

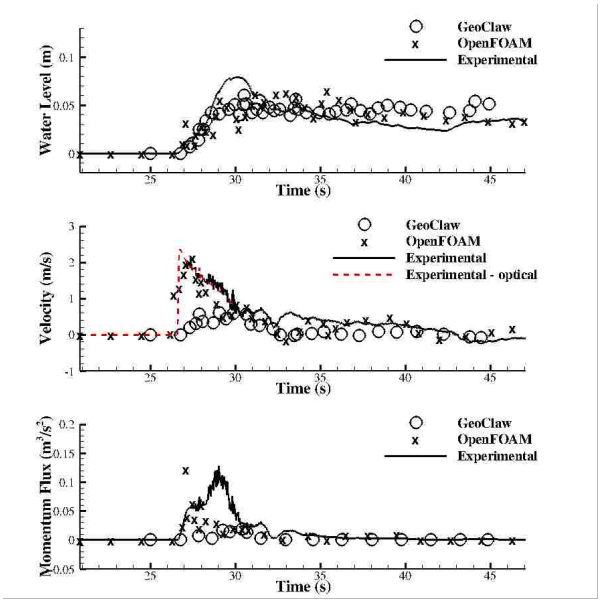


(d) Gauge C8

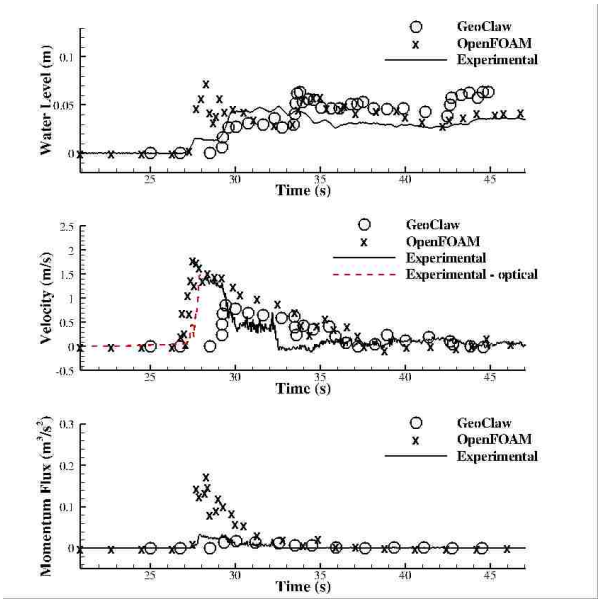
Figure 4.7: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges along line C



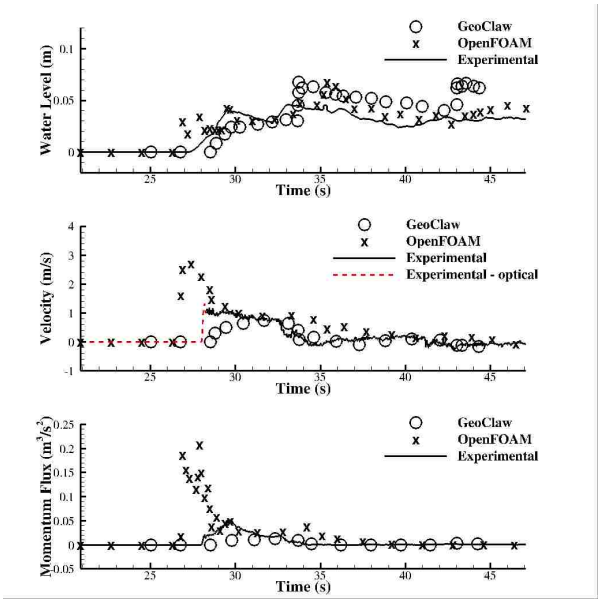
(a) Gauge D1



(b) Gauge D2



(c) Gauge D3



(d) Gauge D4

Figure 4.8: Time histories of surface elevation, cross-shore velocity and momentum flux at some selected gauges in group D

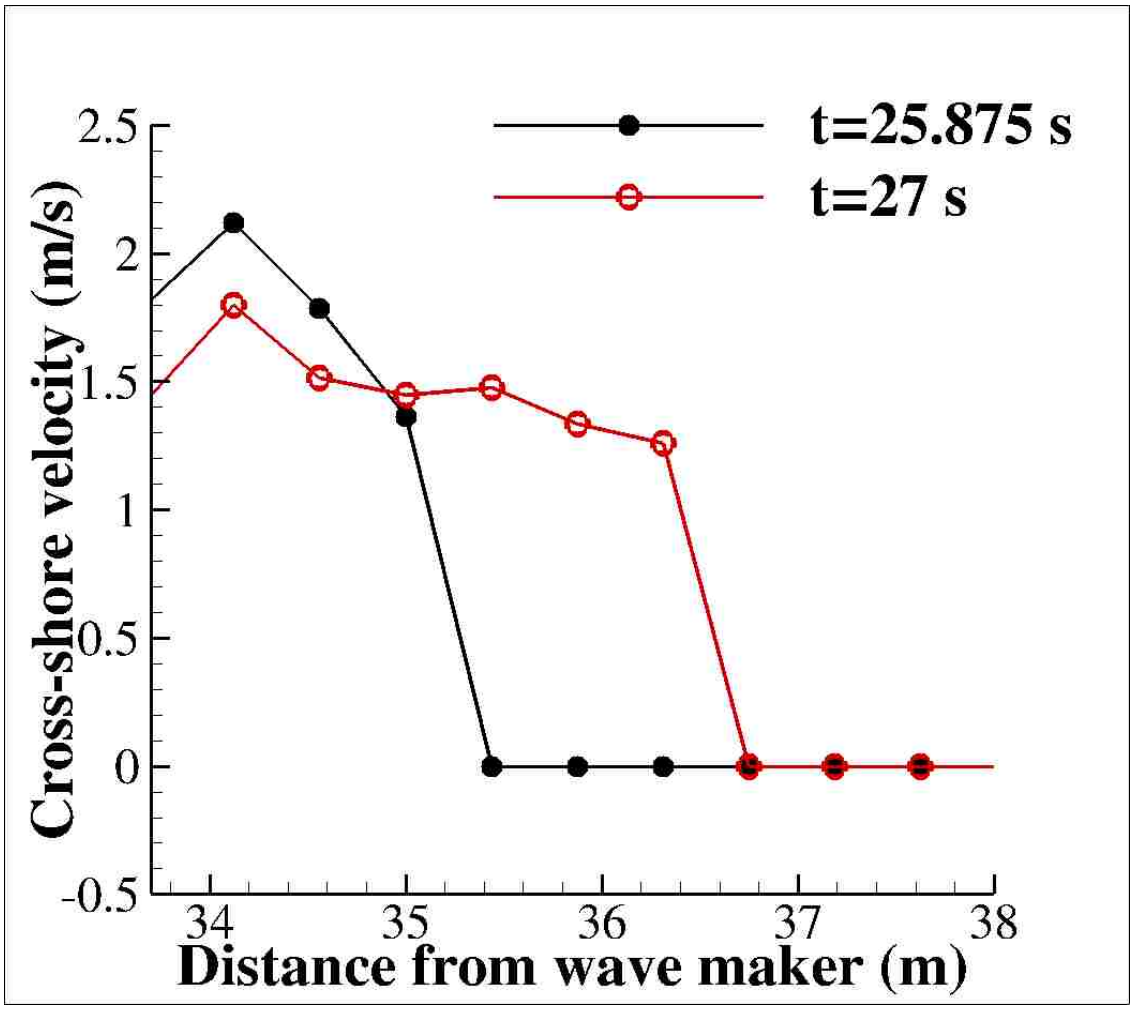


Figure 4.9: Velocity distribution in the bore near gauge A4, from the GeoClaw model

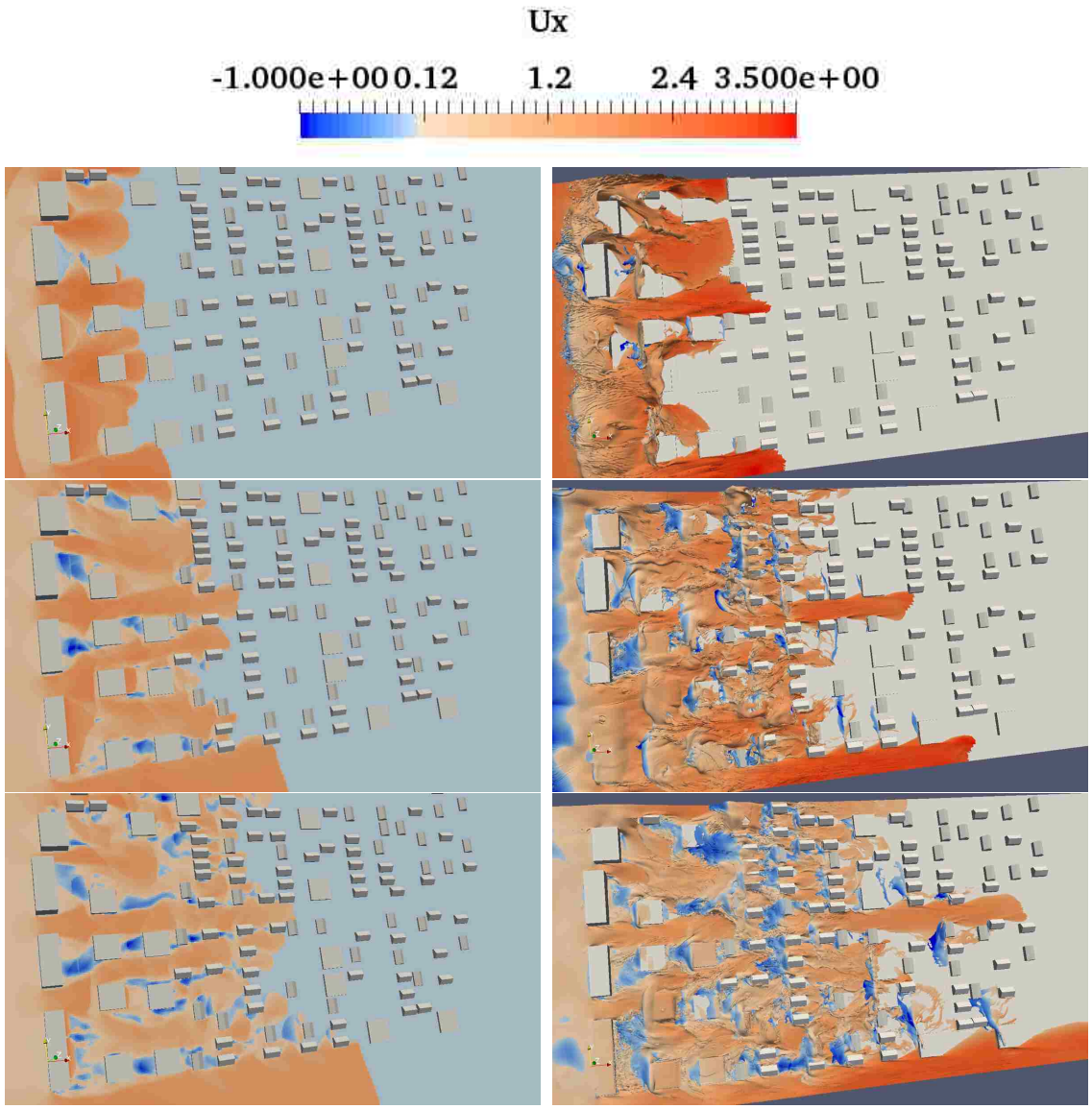


Figure 4.10: Snapshots of the simulation near line A, colored by cross-shore velocity, at 3 different times (from top to bottom): $t = 25.9$ s, $t = 27$ s, t=28.1 s. Left: Geoclaw; Right: OpenFOAM.

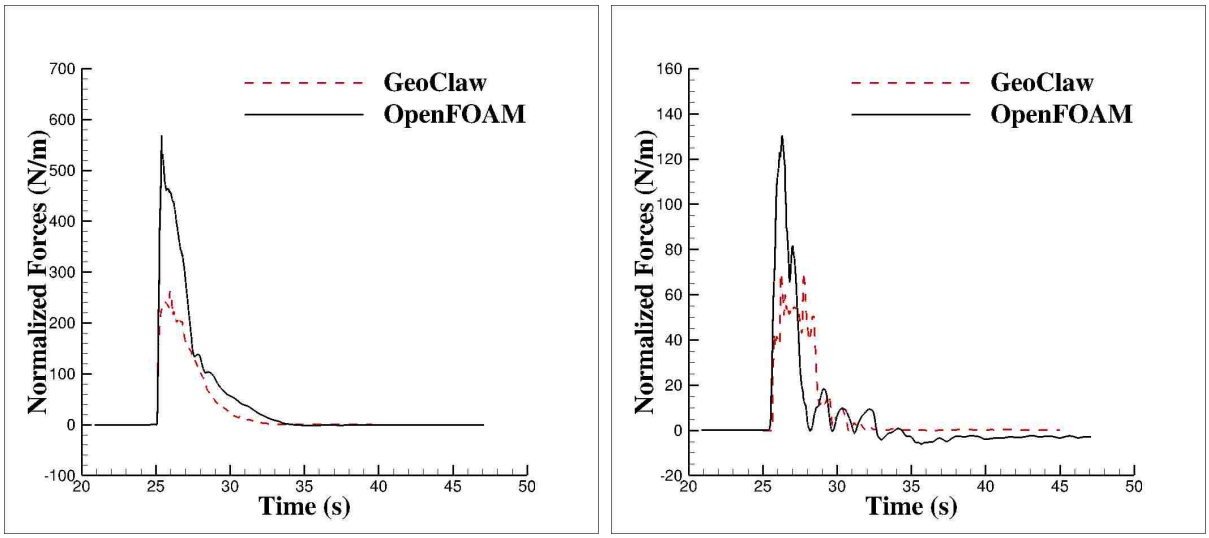
4.3.4 Force Prediction from the Two-Dimensional Model

The 2D model cannot predict forces on structures directly. The same approach in section 3.4 is used here to compute forces on the same six representative buildings from flow parameters (C_d chosen as 2.0 as well): only the building being considered is removed and a case is run to get flow parameters at the center of that building, which minimizes the influence of removing that building on the flow overall.

Figure 4.11 compares predicted forces in the cross-shore direction from the two models on selected buildings. Note that these forces are also normalized by the width of western (left) wall of the buildings.

Peak values of forces predicted by the GeoClaw model on all buildings are only approximately half of those predicted by the OpenFOAM model, except for building V. This is consistent with smaller peak values in the prediction of momentum flux from the GeoClaw model at most of the 31 gauges since both water level and cross-shore velocity are underestimated. For example, as shown in figure 4.5, peak values in momentum flux predicted by the GeoClaw model are approximately half of those predicted by the OpenFOAM model.

Note that, however, prediction of forces from the GeoClaw model becomes better when compared to the OpenFOAM model after the initial impact. This indicates the GeoClaw model's limited ability to capture details of transient interaction between fluids and structures occurs during the initial impact, which is the most important to tsunami hazard assessment in many scenarios, but as the flow begins to interact more with the surrounding coastal infrastructure as the water travels onshore, these strong impact forces may be mitigated. The underestimation of peak forces in figure 4.11, however, indicates that to predict tsunami forces on buildings in coastal communities with the current GeoClaw model, a drag coefficient of 2.0 may not be sufficient.



(a) Building I

(b) Building III

(c) Building V

(d) Building VI

Figure 4.11: Predicted forces in cross-shore direction on selected buildings (normalized). GeoClaw: forces predicted from the drag coefficient; OpenFOAM: forces predicted directly from pressure field

4.4 Conclusions

In this chapter, a 2D tsunami inundation model of the Seaside experiment is developed and compared against the 3D model developed in chapter 3. The 2D model is first validated by comparing water level, velocity profile and forces on a single column impacted by a bore from a dam-break. Then it is used to predict the free surface elevation, velocity and momentum flux of a tsunami inundation in Seaside, Oregon. The predicted flow parameters agree well with experimental measurements in the post-impact region at most gauges. Near the initial-impact region, the 2D GeoClaw model has more difficulty in capturing transient characteristic of the flow. The 3D OpenFOAM model can solve this challenge better but at the expense of much more computational resources required. This is because the variation in the vertical direction is "eliminated" by the integration in 2D model while all 3D characteristics of the flow as well as turbulence are modeled by the 3D model. Several primary conclusions can be drawn for this chapter:

1. The 3D RANS model can predict flow parameters and forces on structures by modeling subsections of the entire domain without loss of accuracy in areas of interest, while the 2D NSWE model can model the entire basin in one simulation, with much less computational resources than the 3D model. The two-dimension model also agrees well with experimental measurements at most locations considered after the initial impact. The RANS model, however, can provide more details of the flow, especially near the initial impact region.

2. The fluid dynamics in the bore front are transient and turbulent. Thus near the initial impact, prediction of flow parameters and forces is challenging but also the most critical since the flow parameters and forces have maximum value near this point. The 3D RANS model solves this challenge better than the 2D NSWE model but needs much more computational resources.

3. Using the approach recommended by FEMA P-646 to predict fluid forces on structures from the 2D model works well in the simple case of flow around a column, but becomes less reliable in complex constructed environments. Although choosing a drag coefficient of 2.0 is

considered conservative, the 2D model with this value was still seen to significantly underestimate fluid forces (in some cases giving only half of the prediction from the 3D model as discussed in Section 4.3.4) because the 2D model underestimates peak velocities in this complex flow. For this reason, it is recommended that a 3D model should be used to determine the tsunami loads on structures when possible, which eliminates the necessity of choosing a large safety factor when only flow velocity is available from the 2D model as done in Ash [2015].

This chapter presents a 2D numerical model for tsunami inundation and compares it to the 3D model described in chapter 3. Challenges in prediction of flow parameters and forces are revealed and the capabilities of the two numerical models in solving this type of problem are analyzed. Tradeoffs need to be made between the two models due to their different levels of accuracy and required computational resources. The comparisons and discussion in this chapter provide a reference when choosing between a 2D model and a 3D model to predict flow parameters and forces on structures in tsunami inundation.

Part II

ACCELERATING THE GEOCLAW MODEL FOR MULTI-SCALE TSUNAMI MODELING USING GRAPHICS PROCESSING UNITS (GPUS)

Chapter 5

INTRODUCTION

5.1 Motivation

The nonlinear shallow water equations (NSWE) are the governing equations for many two-dimensional (2D) tsunami models. Solving the NSWE efficiently reduces the running time of these 2D tsunami models, which is critical to the study of natural hazards such as tsunami and storm surge, since it provides more response time in an early warning system and allows more runs to be done for Probabilistic Tsunami Hazard Assessment (PTHA) where thousands of runs may be required. In a tsunami early warning system, tsunami models are used to simulate tsunami propagation and inundation in real time, after a seismic signal is detected. The simulation results provide arrival time and severity of potential tsunami hazards at locations of interest before the arrival of the tsunami. The faster these tsunami models can run, the earlier this system can send out the warnings and the more response time emergency managers, citizens and first responders in coastal regions can have. In PTHA, high-fidelity hazard curves and hazard maps are produced to provide information like the probability of the maximum flooding depth at a location exceeding a certain value of interest. To construct these curves and maps, thousands of tsunamis are simulated, either from pre-designed representative scenarios, each associated with a certain probability, or from tsunami events parameterized by inputs sampled from some random parameter space. A fast tsunami model allows more such tsunami simulations to be conducted within a certain time budget and thus generates higher-fidelity PTHA outcomes.

This chapter presents an efficient CUDA implementation of the Geoclaw software used in chapter 4, motivated by the need for having a fast tsunami model. The open source GeoClaw code is distributed as part of Clawpack [Clawpack Development Team], which is widely used for modeling tsunamis and has undergone a benchmarking process in 2011 [Horrillo et al., 2014] as part

of a National Tsunami Hazard Mitigation Program (NTHMP), allowing its use on tsunami hazard modeling projects supported by this program, [e.g., Titov et al., 2018, González et al., 2013]. It has been used for many other projects involving tsunami hazard assessment and/or the study of historic or paleo tsunamis, [e.g., Arcos and LeVeque, 2015, Amir et al., 2013, Cienfuegos et al., 2018, Galanti et al., 2011, Hayes and Furlong, 2010, Johnstone and Lence, 2009, MacInnes et al., 2013, Ren et al., 2013]. GeoClaw has also been used in modeling storm surge (e.g. Mandli and Dawson [2014]) and dam failures (e.g. George [2011], Turzewski et al. [2019]), and so the GPU-accelerated version should prove useful beyond tsunami modeling.

The CPU version of GeoClaw is parallelized with OpenMP on multicore shared memory machines, and this part of the dissertation describes how this has been extended to use CUDA-based GPU acceleration on such a machine [Nickolls et al., 2008]. As the original GeoClaw cannot run on distributed memory machines, using the GPU to accelerate the computation within a single compute node can be an efficient way of improving performances. Nodes of this nature are available at many supercomputer centers and cloud platforms, as well as on many desktop and laptop machines. Due to the use of adaptive mesh refinement (AMR) in GeoClaw, many realistic modeling problems can be solved on such hardware without the need for distributed memory parallelization. The GPU acceleration allows an additional increase in speed that will be particularly useful for tsunami early warning system and PTHA applications where many simulations must be conducted.

The GPU code developed in this study can be found on Github at `https://github.com/xinshengqin/geoclaw/tree/geo_gpu_paper` and on Zenodo at `https://doi.org/10.5281/zenodo.2727368`. It is also being freely incorporated into the Clawpack package [Clawpack Development Team]. More details can be found on `www.clawpack.org/gpu.html`.

5.2 Related Work

As discussed above, being able to run a tsunami model fast can benefit a tsunami early warning system and PTHA study significantly. More generally, the speed of a numerical model can be

increased by either reducing computational cost, using better hardware, or both.

5.2.1 Reducing Computational Cost with Adaptive Mesh Refinement

Since being proposed by Berger and Oliger [1984], the AMR algorithm has been shown to effectively reduce computational cost in the numerical simulation of multi-scale problems. It can track features much smaller than the overall scale of the problem and adjust the computational grid during the simulation. The algorithm has been implemented and developed into several frameworks and can be categorized into three major variants. The first one is often referred to as patch-based or structured AMR [Berger and Colella, 1989]. It allows rectangular grid patches of arbitrary size and any integer refinement ratios between two level of grid patches [e.g., Hornung and Kohn, 2002, Bryan et al., 2014, Clawpack Development Team, Zhang et al., 2016, Adams et al., 2015]. Another variant is the cell-based AMR, which refines individual cells and often uses a quadtree or octree data structure to store the grid patch information. The last variant is a combination of the first two, often referred to as block-based AMR. Unlike the patch-based AMR, which stores the multi-resolution grid hierarchy as overlapping and nested grid patches, this approach stores the grid hierarchy as non-overlapping fixed-size grid patches, each of which is stored as a leaf in a forest of quadtrees or octrees [e.g., Burstedde et al., 2014, 2011, Fryxell et al., 2000, MacNeice et al., 2000]. In the past two decades, the AMR algorithm has been extensively applied for geophysical applications [e.g., Leng and Zhong, 2011, Burstedde et al., 2013, LeVeque et al., 2011]. In particular, tsunami models that simulate both large scale transoceanic tsunami propagation and inundation of small-scale coastal regions save several orders of computational cost by using AMR.

5.2.2 Faster Simulation with Better Hardware

Another approach to increasing the execution speed of a numerical model is to use better hardware and/or a parallelize the code to take advantage of modern architectures. Several codes parallelize tsunami modeling on multi-core CPUs [e.g., Pophet et al., 2011], and GeoClaw takes this approach via OpenMP. ForestClaw [Burstedde et al., 2014], on the other hand, can simulate a tsunami on

distributed-memory machines with MPI parallelism. In the past decade, the increase of computing power of CPUs has slowed down due to the breakdown of Moore's law, which states that the number of transistors in a dense integrated circuit doubles approximately every two years [Brock, 2006]. In contrast, leveraging the ability of Graphics Processing Units (GPUs) to do massively paralleled work has become increasingly popular in the community of computational science and engineering. Many researchers from different disciplines such as molecular dynamics (Anderson et al. [2008], Liu et al. [2007]), computational biology (Schatz et al. [2007]), weather forecasting (Michalakes and Vachharajani [2008]) and linear algebra (Barrachina et al. [2008]) have developed numerical models that run on the GPUs to make use of the computing power.

In the tsunami research community, some researchers have reported decent speed-ups in simulating tsunamis by using the GPUs. However, most of these earlier studies involved implementing the PDE solvers on grids with constant spatial resolution (did not use AMR). Earlier, the researchers solved the NSWE on the GPU before the appearance of libraries for general purpose GPU computing like CUDA, when researchers had to write their programs in a way such that they could make use of the GPUs' ability of computing color for a large number of pixels in parallel to update solutions in several grid cells concurrently (Hagen et al. [2005] and Lastra et al. [2009]). Hagen et al. [2005] reported their simulations on the GPU can be 15 to 30 times faster than running on a single CPU. Lastra et al. [2009] achieved ~ 100 and ~ 200 speed-ups for the two different GPUs used and for cases without wet/dry cells. For cases where wet/dry fronts appear near a coastal line, the speed-up decreased but was still up to ~ 80 and ~ 150.

The appearance of programming models for general purpose GPU computing such as CUDA (Nickolls et al. [2008]) makes developing numerical models on the GPUs much easier. Castro et al. [2011] reviewed their work of accelerating NSWE for simulating shallow flows with the GPUs on both structured and unstructured mesh. The NVIDIA GeForce GTX 260 GPU and GeForce GTX 280 they used for testing the structured mesh gave ~ 22 and ~ 25 speed-ups respectively (all versus one CPU core). For the unstructured mesh, the NVIDIA GeForce GTX 260 GPU gave a lower speed-up of around 13 while a more powerful GPU, NVIDIA GeForce GTX 480, showed a speed-up of ~ 40. Smith and Liang [2013] also reported a new software that accelerates the

2D shallow-flow simulation. Their code used a second-order accurate Godunov-type MUSCL-Hancock scheme with an HLLC Riemann solver. The speed-ups were 6.7 for a NVIDIA Tesla M2075 GPU versus an Intel Xeon E5-2609 2.40 GHz quad-core CPU (all four cores used) and 4.2 for an AMD FirePro V7800 GPU versus the same CPU. Lacasta et al. [2015] tested and evaluated a GPU-accelerated 2D NSWE solver on unstructured meshes by simulating a rainfall/runoff event. The test showed 13.47 and 32.07 speed-ups for a NVIDIA Tesla c2075 GPU and a NVIDIA Titan Black GPU respectively, versus a last generation Intel Core i7 4770 @3.40 GHz CPU. Other relevant studies include those of Acuña and Aoki [2009], De La Asunción et al. [2011], Brodtkorb et al. [2012], de la Asunción et al. [2013], and De La Asunción et al. [2016].

5.2.3 Can We Utilize Both Approaches?

A tsunami model that is parallelized on better hardware but does not use AMR (such as models surveyed in section 5.2.2) might be too computationally expensive if a transoceanic tsunami propagation needs to be modeled. Can we utilize both approaches introduced above? This means we need to accelerate an AMR code on the GPU. The complexity of the AMR algorithm and data structure add challenges to the task. There have been some implementations of AMR algorithms on the GPUs with application to astrophysics. Wang et al. [2010] implemented compressible inviscid fluid solvers with block-structured AMR on the GPU using NVIDIA's CUDA programming model. The efficiency of their code were showed by comparing its execution time on a Quadro FX 5600 GPU and a single 3 GHz CPU core. They first showed that their solver could achieve ~ 10 speed-up on uniform grids for a large range of grid size (32^3 to 256^3) since with AMR, a wide variety of grid sizes can arise. After that, they simulated a 3D cloud disruption case with AMR, which still got a speed-up of ~ 8, to show the efficiency of their code even with complex grid hierarchy in AMR. Schive et al. [2010] described a GPU accelerated AMR code called GAMER for fluid simulation in astrophysics. The hydrodynamic solver used a directional split relaxing total variation diminishing scheme to solve the Euler Equations and the solver for Poisson Equations use a Fast Fourier Transfer (FFT) method and the successive overrelaxation (SOR) method. Schive et al. [2011] further added some directional unsplitting methods on the GPU for the hydrodynamic

solver in GAMER. However, very few models for simulating tsunamis with AMR on the GPU have been developed and this is the goal of this part of the dissertation. One relevant work is the simulation of landslide-generated tsunamis on the GPUs by de la Asunción and Castro [2017] for example.

5.3 Overview

Chapter 6 summarizes the AMR and finite volume methods implemented in GeoClaw to the extent needed to explain the GPU implementation, which is described in 7. Many more details can be found in previous publications; in particular LeVeque et al. [2011] give an extensive discussion of the algorithms, along with an overview of tsunami modeling applications, while Berger et al. [2011] discuss more details of the software, with additional test problems. More references are summarized in table 5.1.

Chapter 7 presents a CUDA implementation of the patched-based AMR algorithm in GeoClaw, which has been developed and used to simulate tsunamis on the GPU. The Godunov-type wave-propagation scheme with 2nd-order limiters is implemented to solve the nonlinear shallow water system with varying topography. Both canonical Cartesian grid coordinates for modeling tsunamis in small regions and spherical coordinates for transoceanic tsunami propagation are supported. The use of AMR adds challenges to the implementation, including dynamic memory structure creation and manipulation, balanced distribution of computing loads between the CPU and the GPU, and optimizations to minimize global memory access and maximize arithmetic efficiency in the GPU kernel.

Chapter 8 begins with the introduction to three metrics that are proposed to evaluate the absolute performance of the model, which shows efficient usage of hardware resources. Two numerical experiments on the 2011 Japan tsunami and a local tsunami triggered by a hypothetical Mw 7.3 earthquake on the Seattle Fault are then presented to illustrate the correctness and efficiency of the GPU code, which implements a simplified dimensionally-split version of the algorithms. Both numerical simulations are conducted on sub-regions on a sphere with adaptive grids that adequately resolve the propagating waves. The GPU implementation, when running on a single GPU, is ob-

Table 5.1: Some references for different modules in the GeoClaw code.

Module in GeoClaw	**References**
Overall introduction	Berger et al. [2011], LeVeque et al. [2011]
Wave propagation algorithm	LeVeque [1997]
The regridding process	Berger and Rigoutsos [1991]
The refluxing process	Berger and LeVeque [1998]
The interpolation and averaging process	LeVeque et al. [2011]
The augmented shallow water equations and riemann solver	George [2006, 2008]

served to be 3.6 to 6.4 times faster than the original model running in parallel on a 16-core CPU. The chapter ends with conclusions in section 8.3.

Chapter 6

METHODOLOGY

The nonlinear shallow water equations (NSWE) solved by GeoClaw have been introduced in section 4.1 and thus are not repeated here. This chapter focuses on introducing the numerical methods used in GeoClaw for solving the NSWE and the AMR algorithm.

6.1 Finite Volume Methods, Wave Propagation Algorithm and Riemann Solvers

Consider a one-dimensional homogeneous system of hyperbolic equations in conservative form:

$$q_t + f(q)_x = 0, \tag{6.1}$$

where $q(x,t) \in \mathbb{R}^m$ is a vector of m components representing the unknowns, and $f(q)$ is a m-dimensional vector of flux functions. The subscripts t and x represent partial derivatives with respect to time t and space x respectively. In a finite volume method in one-dimensional space, the spatial domain is subdivided into intervals (often referred to as grid cells). The approximation to the integral of q over each of these cells is tracked at each time step. Such an approximation at time t_n in the ith cell $\mathcal{C}_i = \left[x_{i-1/2}, x_{i+1/2}\right]$ is:

$$Q_i^n = \frac{1}{V_i} \int_{x_{i-1/2}}^{x_{i+1/2}} q(x, t_n)\, dx \tag{6.2}$$

where V_i is the volume of cell $\mathcal{C}_i$ (in this case $V_i = x_{i+1/2} - x_{i-1/2}$). The wave propagation algorithm updates the ith cell in time by solving Riemann problems at $x_{i-1/2}$ and $x_{i+1/2}$ and using the resulting wave of the two Riemann problems to determine the numerical update:

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}^n - F_{i-1/2}^n\right), \tag{6.3}$$

where $F^n_{i-1/2}$ a numerical flux that approximates the time average of the true flux at the left boundary of cell $\mathcal{C}_i$, $x_{i-1/2}$, during the time interval $[t_n, t_{n+1}]$:

$$F^n_{i-1/2} \approx \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f\left(q\left(x_{i-1/2}, t\right)\right)\, dt. \tag{6.4}$$

The Godunov-type method chooses $q(x_{i-1/2}, t) = Q_{i-1/2}$ in equation (6.4) so the numerical flux is computed as

$$\begin{aligned} F^n_{i-1/2} &= \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} f\left(Q_{i-1/2}\right)\, dt \\ &= f\left(Q_{i-1/2}\right), \end{aligned} \tag{6.5}$$

where $Q_{i-1/2} := Q\left(x_{i-1/2}\right)$ is the solution state at $x_{i-1/2}$ determined from solving the Riemann problem there at t_n.

6.1.1 Wave Propagation Algorithm

Equation (6.3) has a nice property that no matter how the numerical fluxes are chosen, the solution is conservative in the domain (with exception at the boundaries). However, the Riemann solver used in the current implementation for solving nonlinear shallow water equations incorporates the source term $-ghB_x$ and $-ghB_y$ on the right hand side of equation (4.1), which converts the nonlinear shallow water equations to non-conservative form. As a result, the conservative form (6.3) cannot be used. A one-dimensional homogeneous system of hyperbolic equations in non-conservative form can be written as:

$$q_t + A(q)q_x = 0, \tag{6.6}$$

For non-conservative form, the wave propagation algorithm can be used to update the solution, which replaces equation (6.3) with:

$$Q^{n+1}_i = Q^n_i - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+ \Delta Q_{i-1/2} + \mathcal{A}^- \Delta Q_{i+1/2}\right), \tag{6.7}$$

where $\mathcal{A}^+ \Delta Q_{i-1/2}$ is the net effect of all right-going waves propagating into cell $\mathcal{C}_i$ from its left boundary, and $\mathcal{A}^- \Delta Q_{i+1/2}$ is the net effect of all left-going waves propagating into cell $\mathcal{C}_i$ from its

right boundary. Namely,

$$\mathcal{A}^+\Delta Q_{i-1/2} = \sum_{p=1}^{m} (\lambda^p)^+ \mathcal{W}^p_{i-1/2}, \tag{6.8a}$$

$$\mathcal{A}^-\Delta Q_{i+1/2} = \sum_{p=1}^{m} (\lambda^p)^- \mathcal{W}^p_{i+1/2}, \tag{6.8b}$$

where m is total number of waves, $\mathcal{W}^p$ is the pth wave from the Riemann problem, λ^p is wave speed of the pth wave, and

$$(\lambda^p)^+ = max(\lambda^p, 0), \quad (\lambda^p)^- = min(\lambda^p, 0). \tag{6.9}$$

The notations here are motivated by the linear case where $f(q) = Aq$. In such a case, the waves are simply decomposition of the initial jumps into basis form by the eigenvectors of the coefficient matrix A, propagating at the speed of eigenvalues:

$$q_r - q_l = \sum_{p=1}^{m} \mathcal{W}^p = \sum_{p=1}^{m} \alpha^p r^p, \tag{6.10}$$

where q_r and q_l are right and left states of the Riemann problem, r^p is the pth eigenvector of matrix A, and α^p is coordinate in the direction of r^p.

6.1.2 Second-Order Corrections and Wave Limiters

The wave propagation form of Godunov's method (equation (6.7)) is only first-order accurate and introduces a great amount of numerical diffusion into the solution. This often smears out the steep gradients in the solution which are common in surface elevation near shoreline in tsunami simulation. To obtain second-order resolution and maintain steep gradients, additional terms are added to equation (6.7):

$$\begin{aligned} Q_i^{n+1} = Q_i^n &- \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+\Delta Q_{i-1/2} + \mathcal{A}^-\Delta Q_{i+1/2}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\tilde{F}^n_{i+1/2} - \tilde{F}^n_{i-1/2}\right). \end{aligned} \tag{6.11}$$

The second-order correction terms are computed as

$$\tilde{F}^n_{i-1/2} = \frac{1}{2}\sum_{p=1}^{m}\left(1 - \frac{\Delta t}{\Delta x}|\lambda^p|\right)|\lambda^p|\widetilde{\mathcal{W}}^p_{i-1/2}, \tag{6.12}$$

where the time step index n is dropped and the superscript p refers to the wave family. The wave $\widetilde{\mathcal{W}}^p_{i-1/2} = \Phi(\theta^p_{i-1/2})\mathcal{W}^p_{i-1/2}$ is a limited version of the original wave $\mathcal{W}^p_{i-1/2}$, where $\theta^p_{i-1/2}$ is a scalar that measures the strength of wave $\mathcal{W}^p_{i-1/2}$ relative to waves in the same wave family arising from a neighboring Riemann problem:

$$\theta^p_{i-1/2} = \frac{\mathcal{W}^p_{I-1/2} \cdot \mathcal{W}^p_{i-1/2}}{\|\mathcal{W}^p_{i-1/2}\|}, \tag{6.13}$$

where the index I represents the interface on the upwind side of interface $x_{i-1/2}$

$$I = \begin{cases} i-1, & \text{if } \lambda^p_{i-1/2} > 0, \\ i+1, & \text{if } \lambda^p_{i-1/2} < 0, \end{cases} \tag{6.14}$$

$\Phi(\theta)$ is a limiter function that gives values near 1 where solution is smooth and is close to 0 near discontinuities. Such property of a limiter function preserves second-order accuracy in region where the solution is smooth while avoiding non-physical oscillations arising near the discontinuities. The MC limiter function used in all the benchmarks in this study is:

$$\Phi(\theta) = max(0, min(\frac{1+\theta}{2}, 2, 2\theta) \tag{6.15}$$

6.1.3 The Riemann Solver for the Shallow Water Equations

As discussed in previous sections, solving the Riemann problems at cell interfaces is the basis for updating the solutions in time. In fact, the cost of solving the Riemann problems often dominates the computational cost of the entire time integration procedure. The Riemann solver used in the current implementation is an approximate Riemann solver, which returns a set of waves that are simple discontinuities propagating at constant speeds, whereas the exact Riemann solution might include rarefaction waves with varying speeds.

The one-dimensional shallow water equations can be written as

$$h_t + \mu_x = 0, \tag{6.16a}$$

$$\mu_t + \phi_x + ghB_x = 0, \tag{6.16b}$$

where $\mu = hu$ is flux of h, $\phi = hu^2 + \frac{1}{2}gh^2$ is flux of momentum. These equations are augmented to a system of four equations for the Riemann solver and are written in non-conservative form as equation (6.6), with

$$q = \begin{bmatrix} h \\ \mu \\ \phi \\ B \end{bmatrix}, A(q) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ gh - u^2 & 2u & 0 & gh \\ 0 & gh - u^2 & 2u & 2ghu \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.17}$$

The equation for the momentum flux ϕ can be derived by differentiating $\phi = \frac{\mu}{h} + \frac{1}{2}gh^2$ with respect to t. Introducing this extra component ϕ allows a more accurate approximation to Riemann problems with a large rarefaction and a natural entropy fix for transonic rarefactions [George, 2008]. By viewing the topography $B(x,t)$ as a function of space x and time t that does not vary with time, the source term is eliminated, though we still need to deal with the source term introduced by friction force and/or other driving forces. Including the topography B into the Riemann solver is critical in preserving steady state of the ocean even on very coarse grids, which often exists in AMR.

To solve this non-conservative system, an approximate Riemann solver was developed by George [2006, 2008, 2011]. Here, only some of the important properties of this Riemann solver that are critical to tsunami modeling are summarized as below:

- It gives a good approximation to the exact Riemann solution.
- It is well-balanced. Namely, it preserve the state of ocean at rest.
- It can handle dry states in the Riemann problem (left or right state of the Riemann problem has water depth $h = 0$).
- It is depth positive semidefinite — the water depth will not be negative in any case.
- It works well in conjunction with AMR.

6.1.4 *Source Terms and Time Step Restrictions*

Although the source term introduced by gradient of the topography B is included in the Riemann solver, and thus does not appear as a source term, additional source terms arise from bottom friction in shallow water (the second term in the right hand side of equation (4.1b) and (4.1c). This is handled through operator splitting in the current implementation. Use a one-dimensional hyperbolic system as an example. $q_t + f(q)_x = \psi(q, x)$ can be handled by alternatively solving $q_t + f(q)_x = 0$ with the method introduced in the previous sections, and $q_t = \psi(q, x)$, which is simply a independent system of Ordinary Differential Equations (ODEs).

The time step size Δt for time integration must be chosen and adapted carefully at each time step if variable time step is used, which is typical for tsunami modeling. The Courant, Friedrichs and Lewy (CFL) condition implies that the time step size for a certain AMR level must satisfy

$$\nu \equiv \left| \frac{s\Delta t}{\Delta x} \right| \leq 1 \tag{6.18}$$

where ν is the CFL number and s is the maximum wave speed seen on the grid.

6.1.5 *Algorithms in Two-Dimensional Space*

A two-dimensional hyperbolic system in non-conservative form

$$q_t + A(q)q_x + B(q)q_y = 0 \tag{6.19}$$

is a general extension of the one-dimensional hyperbolic system (equation (6.6)) in two-dimensional space.

The Godunov-type finite volume algorithms discussed above can be naturally extended to two-dimensional space by dimensional splitting, which splits the two-dimensional problem into a sequence of one-dimensional problems. The wave propagation algorithm now becomes

$$\begin{aligned} Q^*_{ij} = Q^n_{ij} &- \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+ \Delta Q_{i-1/2,j} + \mathcal{A}^- \Delta Q_{i+1/2,j}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\tilde{F}^n_{i+1/2,j} - \tilde{F}^n_{i-1/2,j}\right), \end{aligned} \tag{6.20}$$

$$
\begin{aligned}
Q_{ij}^{n+1} = Q_{ij}^{*} &- \frac{\Delta t}{\Delta y}\left(\mathcal{B}^{+}\Delta Q_{i,j-1/2} + \mathcal{B}^{-}\Delta Q_{i,j+1/2}\right) \\
&- \frac{\Delta t}{\Delta y}\left(\tilde{G}_{i,j+1/2}^{n} - \tilde{G}_{i,j-1/2}^{n}\right),
\end{aligned}
\tag{6.21}
$$

where $\mathcal{A}^{+}\Delta Q_{i-1/2,j}$ and $\mathcal{A}^{-}\Delta Q_{i+1/2,j}$ are net effect of all waves propagating into cell $\mathcal{C}_{ij}$ from its left and right edges, while $\tilde{F}_{i-1/2,j}^{n}$ and $\tilde{F}_{i+1/2,j}^{n}$ are 2nd-order correction fluxes through its left an right edges. Equation (6.20) is essentially updating solution of the one-dimensional hyperbolic system

$$q_t + A(q)q_x = 0 \tag{6.22}$$

Similarly, $\mathcal{B}^{+}\Delta Q_{i,j-1/2}$ and $\mathcal{B}^{-}\Delta Q_{i,j+1/2}$ are net effect of all waves propagating into cell $\mathcal{C}_{ij}$ from bottom and top edges, while $\tilde{G}_{i,j-1/2}^{n}$ and $\tilde{G}_{i,j+1/2}^{n}$ are 2nd-order correction fluxes through its bottom and top edges. Equation (6.21) is essentially updating solution of the one-dimensional hyperbolic system

$$q_t + B(q)q_y = 0 \tag{6.23}$$

6.2 Adaptive Mesh Refinement

The block-structured AMR algorithm is used in the 2D model used in chapter 4, which is built on the GeoClaw software. A brief introduction of the algorithm implemented in GeoClaw is described in this section. Details can be found in many other papers, including Berger and Oliger [1984] and Berger and Colella [1989].

In AMR, a collection of rectangular grid patches are used to store the solution. Grid patches at different levels have different cell sizes. The coarsest grid patches (level 1) cover the entire domain. Grids patches at level l+1 are finer than coarser level l grid patches by integer refinement ratios r_x^l and r_y^l in the two spatial directions, $\Delta x^{l+1} = \Delta x^l / r_x^l, \Delta y^{l+1} = \Delta y^l / r_y^l$, and cover sub-region of level l grid patches. In this study, the refinement ratios in the two spatial directions are always taken to be equal, $r_x^l = r_y^l$. Typically, the time step size is also refined the same factor for level l+1 grid patches, $\Delta t^{l+1} = \Delta t^l / r_t^l$, with $r_t^l = r_x^l = r_y^l$.

The high level grid patches are regenerated every K time steps such that they move with features in the solution. When level l+1 grid patches need to be regenerated, some cells at level l are flagged for refinement based on some criterion (in GeoClaw, typically where the amplitude of the wave is above some specified tolerance, or in specified regions where higher refinement is required, for example near the target coastal location, or where the wave will affect the solution in destination during time interval of interest as indicated by the backward adjoint solution[Davis and LeVeque, 2016]). The flagged cells are then clustered into new rectangular grid patches, which usually include some cells that are not flagged as well, using an algorithm proposed by Berger and Rigoutsos [1991]. The algorithm tries to keep a balance between minimizing the number of grid patches and minimizing the number of unflagged cells that are included in the resulting rectangular grid patches. The newly generated level l+1 grid patches get their initial solution from either copying data from existing old level l+1 grid patches or, if no such grid patch exists, interpolating from level l grid patches. We say the level l+1 patch cells are "on top" of some level l cells that cover the same spatial region. Note that the algorithm described below integrates the underlying level l grid patches before the level $l+1$ patches. After updating the finer patches, any level l cells under level $l+1$ cells have their values updated to the average of level $l+1$ cell values. After each regridding step, the new level $l+1$ patches need not cover the same level l cells as previously.

6.2.1 Time Integration

Each grid patch in the AMR grid hierarchy, despite different resolution, can be integrated in time with the wave-propagation form of Godunov's method described in the previous section. Specifically, the following steps are applied recursively, starting from the coarsest grid patches at level $l = 1$, as illustrated in a simple case in Figure 6.1.

1. Advance the solution in all level l grid patches at t_n by one step of length Δt^l to get solution at $t_n + \Delta t^l$.

2. Fill the ghost cells for all level $l+1$ grid patches, by either copying cell values from adjacent level $l+1$ grid patches if any exists, or interpolating in space and time from the cell values

at level l at t_n and $t_n + \Delta t^l$ if no adjacent level $l+1$ grid patch exists. Note that interpolation in time is generally required because the finer grids are integrated with smaller time steps.

3. Advance the solution at level $l+1$ for r_t^l time steps such that solution at level $l+1$ is at the same time as solution at level l. Each time level $l+1$ is advanced, this entire algorithm (step 1–5) is applied recursively to the next finer level (with l replaced by $l+1$ in these steps) if additional level(s) exist.

4. For any grid cell at level l that is covered by level $l+1$ grid cells, the solution Q in that cell is replaced with an appropriate weighted average of the values from the $r_x^l r_y^l$ level $l+1$ cells on top. This is referred to below as the *updating process*.

5. For any grid cell at level l that is adjacent to level $l+1$ grid cells, the solution Q in that cell is adjusted to replace the value originally computed using fluxes found on level l with the potentially more accurate value obtained by using level $l+1$ fluxes on the side of this cell adjacent to the level $l+1$ patch. This step also preserves conservation for certain problems and is referred to below as the *refluxing process*. (This step is dropped in our implementation, see below.)

Step 5 is important for some problems where exact conservation is expected, e.g., of a conserved tracer or for strong shock waves in nonlinear problems, and is necessary in this case to avoid the use of different numerical fluxes at the same interface on the side of the fine patch and the side where it abuts a coarser grid. However, this step requires storing additional flux information at every time step and communicating this information between levels and was found to have a large negative effect on the ability to speed up the code on the GPU. We also found that this refluxing step has very little effect on the numerical results obtained for tsunami modeling (as shown in Section 8.1). In this application we do not expect conservation of momentum at any rate (due to the topographic and friction source terms) and even conservation of mass is sacrificed when AMR is applied to a cell near the coast (as described in LeVeque et al. [2011]). For these reasons we omit Step 5 in the GPU implementation. This greatly helps to optimize the logistics of the

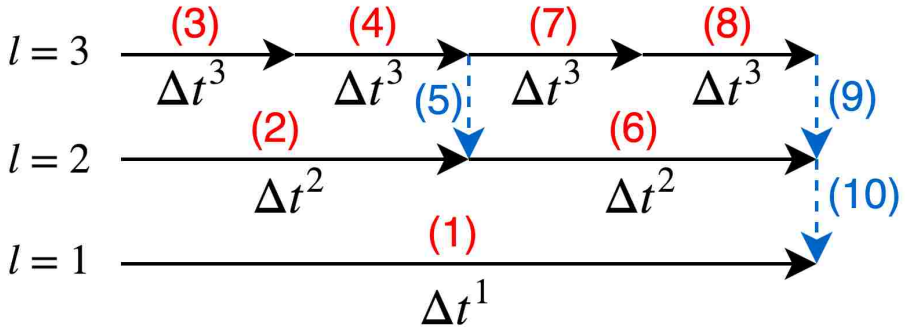


Figure 6.1: Advancing the coarsest level by one time step, for a AMR hierarchy with 3 levels of grid patches. The refinement ratio is 2 for both level $l = 1$ and level $l = 2$. Each black horizontal arrow in a solid line represents taking one time step on a specific level. Each blue vertical arrow in a dashed line represents one updating process that averages the solution from a fine level to a coarse level and refluxing process that preserves global conservation. The numbers from (1) to (10) describe the orders in which all operations are taken.

code and achieve very impressive performance in the benchmark problems, while only introducing negligible changes to the solution.

6.2.2 Regridding

Every time a level is advanced by b time steps, regridding based on this level is conducted (except on the finest allowed level). A larger b results in less frequent regridding, which reduces time spent on the regridding process. However, in order to ensure the waves in the solution do not propagate beyond the refined region before the next regridding process, when cells are flagged for refinement, usually an extra layer of b cells surrounding the original flagged cells are flagged. This makes each grid patch $2b$ cells wider in each of the two horizontal dimensions and thus introduces more cells, which increases the computational time of time integration. As both the regridding process and time integration process contribute to the total running time, a trade-off must be made

in order to lower the total running time when choosing a good value for b, as discussed in Qin et al. [2018a]. However, it is in general hard to determine the "sweet spot" for b as the running time for the regridding process depends on the flagging criteria being chosen while the running time for the time integration process depends on the equations being solved. Typically, b is chosen empirically as 2–4 in tsunami modeling.

In the regridding process, cells must be flagged before they are clustered into new grid patches. A variety of different flagging criteria have been implemented, including flagging based on the slope of the sea surface, sea surface elevation, or adjoint methods. For all the benchmarks in this study, the sea surface elevation is used for flagging. In addition to this, some spatio-temporal regions might also be specified to enforce flagging in these regions, which is very useful for problems where both the transoceanic propagation and local inundation of a tsunami must be modeled, and thus require grid cells that are $O(10^3)$–$O(10^4)$ finer than the coarsest resolution in some near shore regions like a harbor or bay.

During regridding, if a newly generated grid cell cannot copy values from old cells at the same level, its initial value must be interpolated from coarser levels. In the updating process, coarse cell values get updated with the appropriate averaged value of fine grid cells on top. An important requirement for both the interpolation and averaging strategies in tsunami modeling is to maintain the steady state of the ocean at rest, since refinement generally occurs before the tsunami waves arrive in an undisturbed area of the ocean. For areas far from shoreline, the interpolation strategy can be simple linear interpolation and the averaging strategy used is averaging the surface elevation in fine cell values arithmetically and then compute depth in each cell based on the topography and surface elevation. However, near the shoreline where one or more cells is dry, it is impossible to maintain conservation of mass and also preserve the flat sea surface during the interpolation or averaging in some circumstances. Details of the strategies can be found in LeVeque et al. [2011].

6.2.3 Interpolation and Averaging Strategies for Interpolation and Updating Process

During regridding, if a newly generated grid cell cannot copy values from old cells at the same level, its initial value must be interpolated from coarser levels. In the updating process, coarse cell

values get updated with the appropriate averaged value of fine grid cells on top.

An important requirement for the interpolation and averaging strategies in tsunami modeling is to maintain the steady state of the ocean at rest, since refinement generally occurs before the tsunami waves arrive in an undisturbed area of the ocean. For areas far from shoreline, the interpolation strategy can be simple linear interpolation and the averaging strategy used is averaging the surface elevation in fine cell values arithmetically and then compute depth in each cell based on the topography and surface elevation. However, near the shoreline where one or more cells is dry, it is impossible to maintain conservation of mass and also preserve the flat sea surface during the interpolation or averaging in some circumstances. The conservation of mass has to be sacrificed to keep the flat sea surface, since otherwise a tiny resulting gradient in sea surface will generate spurious waves that ruin the solution quickly. The current code has adopted interpolation and averaging strategies that conserve the mass except possibly near the shoreline. The momentum is also conserved when the mass is conserved. Details of the strategies can be found in LeVeque et al. [2011].

6.2.4 Refluxing Process

In the updating process introduced in section 6.2.1, any coarse-grid cell that is covered by fine-grid cells is overwritten by some weighted average of the more accurate fine-grid values using the averaging strategy introduced in section 6.2.3. This potentially causes loss of conservation at the coarse level since a fine-grid cell near find-grid boundary (coarse-fine interface) were updated with the waves from solving Riemann problems between it and a neighboring ghost cell at the same level, while the coarse cell below were updated with different waves that were from Riemann problems between coarse cells. To fix this, a conservation fix must be added to the coarse cell that is under the neighboring fine ghost cell. This is referred to as refluxing process hereafter.

Figure 6.2 shows an example of a fine grid patch nested in a coarse grid patch. In the refluxing process, the coarse-grid cell $\mathcal{C}_{ij}$ must be modified, for instance, by the following three quantities:

1. the difference between the wave at coarse level that propagates into cell $\mathcal{C}_{ij}$ and all waves at

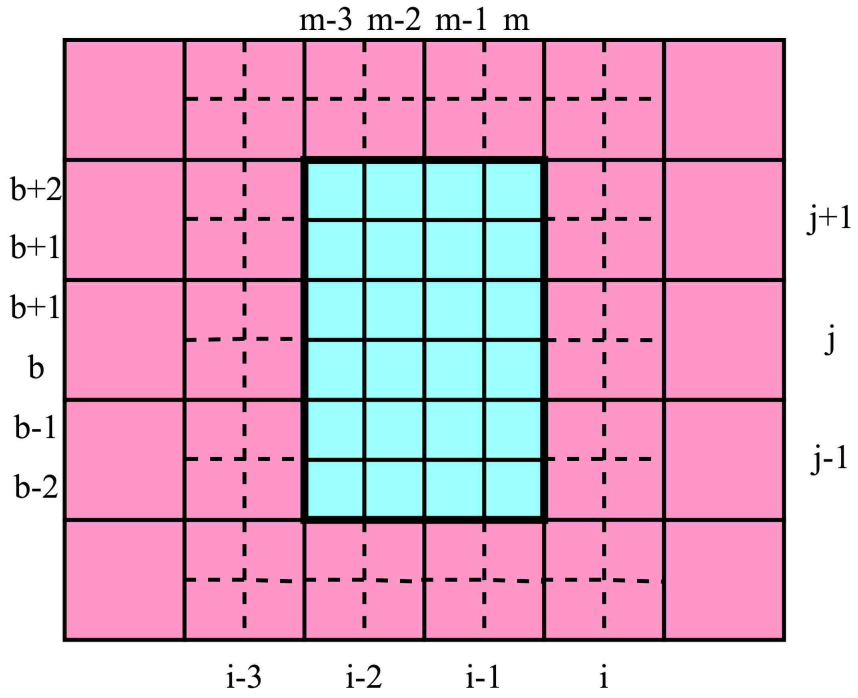


Figure 6.2: One coarse AMR grid patch (pink) and one nested fine AMR grid patch (cyan). The fine grid patch has 4 by 6 internal cells with 2 ghost cells (denoted by dashed lines) on each side. i and j are indices for coarse-grid cells in x and y direction. m and b are indices for fine-grid cells in x and y direction. The refinement ratios in space and time are both 2.

fine level that propagate into cell $\mathcal{C}_{m+1,b}$ and cell $\mathcal{C}_{m+1,b+1}$ within this coarse time step.

2. the waves from solving the Riemann problems between cell $\mathcal{C}_{ij}$ and cell $\mathcal{C}_{m+1,b}$, and between cell $\mathcal{C}_{ij}$ and cell $\mathcal{C}_{m+1,b+1}$, at two intermediate fine time steps.

3. the difference between the second-order correction fluxes through cell edge $x_{i-1/2}$ at the coarse level and those through the same cell edge at the fine level within this coarse time step.

Note that since fine level typically takes many time steps within each coarse-level step, waves and second-order correction fluxes at fine level at the coarse-fine interface at each intermediate step

must be saved. Also, to solve the extra Riemann problem at the coarse-fine interface at each intermediate time step, the relevant coarse-cell values and fine-cell values must be saved. These works, although not as costly as time integration of the solution, contain a relatively large portion of divergent branching along the execution path. Such types of work, if put on the GPU, usually cannot get good performance, due to the fact that each GPU kernel launch has a roughly fixed amount of overhead, and that branching in the GPU kernel code can hurt its performance significantly. On the other hand, performing the refluxing process on the CPU while doing the time integration on the GPU requires either transferring 4 entire arrays of waves (left-going, right-going, up-going, down-going waves) for each grid patch back to the CPU, which is 4 times more expensive than transferring grid solution only, or transferring only the relevant waves near the coarse-fine interface for each grid patch, which are discontiguous in the memory and thus not cheap to transfer. A previous implementation of a similar AMR framework shows the complexity and performance decrease introduced by performing all these operations for refluxing [Qin et al., 2018a]. For these reasons, and given the fact that the mass and momentum are not conservative near the shoreline, the refluxing process is not performed in all the benchmarks in this study. This greatly helps to optimize the logistics of the code, achieve very impressive performance in the benchmark problems, while only introduces negligible inaccuracy to the solution, as shown later in chapter 8.

Chapter 7

CODE DESIGN AND IMPLEMENTATION - A HYBRID CPU/GPU APPROACH

One very basic question to answer in designing a GPU implementation of some code is which part of the program should be done by the CPU and which part should be done by the GPU. In the current implementation we have put the Riemann solvers, wave limiter and CFL reduction on the GPU while letting the CPU take care of the rest, including the updating process, regridding process, filling ghost cells, and updating gauge values (finding the best grid patch to sample quantities of interest from, interpolating from cell values, and output), etc. Since the GPU and the CPU considered in the study have separate physical memory, this design requires the transfer of solution data on each grid patch back and forth between the GPU and CPU memory through a PCI express 2.0 interface, which has relatively low bandwidth compared to the main memory of the CPU and the GPU. However, as it is shown later in the benchmark results, the extra time introduced by such data transfer takes less than 10% of the total running time, since these operations can be carefully hidden by performing other operations concurrently.

If we instead put all procedures on the GPU, although it might save time by eliminating much of the data transfer, the code might suffer from having tiny GPU kernels that add significant overhead to the running time, and running inefficient GPU kernels that can be even slower than the CPU counterpart. One example is filling ghost cells on the GPU. Each GPU kernel that fills ghost cells for a two-dimensional grid patch of a by a cells can have parallelism of $O(a)$ at most ($2a$ ghost cells to fill on each side), which is much less than the parallelism exposed in time integration of the same grid patch, which has parallelism of $O(a^2)$ and often involves much more computation. The overhead of launching a GPU kernel is almost a fixed amount of time regardless of the actual execution time of the kernel. This overhead is often much longer than the execution time of a

tiny GPU kernel like the GPU kernel that would be needed for filling ghost cells. As a result, the total cost (including kernel launching overhead and kernel execution) of doing such operations on the GPU can be even higher than the cost of doing so on the CPU in some cases. In addition to the consideration from kernel launching overhead, code that puts all procedures on the GPU wastes CPU computational resources. Since a typical machine considered in this study consists of a multi-core CPU and a GPU, ideally the work load of the entire program would be distributed between the two as evenly as possible, such that the CPU stays busy as much as possible during the entire execution and so does the GPU. In chapter 8, one metric is proposed to measure such characteristics of a code in numerical experiments.

7.1 Procedure Dependencies and Concurrent Execution

Figure 7.1 shows the major procedures of the code that are hearafter referred to as the *non-AMR portion* of the code. (Omitted are the regridding process and updating process, essential components of AMR.) An arrow from procedure A to procedure B indicates that procedure A must be finished before procedure B can start. The color indicates the type of hardware resource a procedure needs. The four colors represent four major types of hardware resource involved in the execution of the code. A blue block uses one CPU core, a green block uses the GPU Streaming Multiprocessors, a red block uses the memory transfer engine that transfers the data from the CPU memory to the GPU memory, and a purple block uses the memory transfer engine that transfers data in the opposite way. Note that these are separate transfer engines but there is only one of each. Any two procedures without dependency can be done concurrently as long as relevant hardware resources are available. The dependencies in the current implementation are enforced through a combination of rearrangement of CPU procedures and GPU kernel launches, use of OpenMP directives and CUDA streams, and proper synchronization between CPU threads and between the CPU and the GPU.

Figure 7.2 shows an example of these procedures being processed concurrently by four types of hardware on a machine with a three-core CPU. Procedures follow the dependency specified in figure 7.1. Procedures that use the same hardware resource must wait in queue for the hardware to

become available. Note that procedures that needs CPU cores can use any available CPU core so those processed by different CPU cores can be executed concurrently.

In figure 7.2, during the entire time period when the red block for grid patch 12 is processed by one of the two memory transfer engines, the GPU Streaming Multiprocessors are processing the green block for grid patch 14. In this case, transferring the solution data of grid patch 12 from the CPU memory to the GPU memory does not induce any extra cost. During some middle time period of the red block for grid 14, however, no GPU computation is conducted so the GPU Streaming Multiprocessors are idle, which can be caused by unavailability of data on the GPU, for instance. This time segment does induce extra cost due to transferring data between the CPU memory and the GPU memory. Later in chapter 8, such extra cost will be quantified to reveal the influence of transferring data between the CPU memory and the GPU memory on performance of the code. Two additional metrics will also be defined and measured later in chapter 8, the proportion of time during which the CPU has some work to do instead of waiting for the GPU to finish, and the proportion of time during which GPU Streaming Multiprocessors are doing computations.

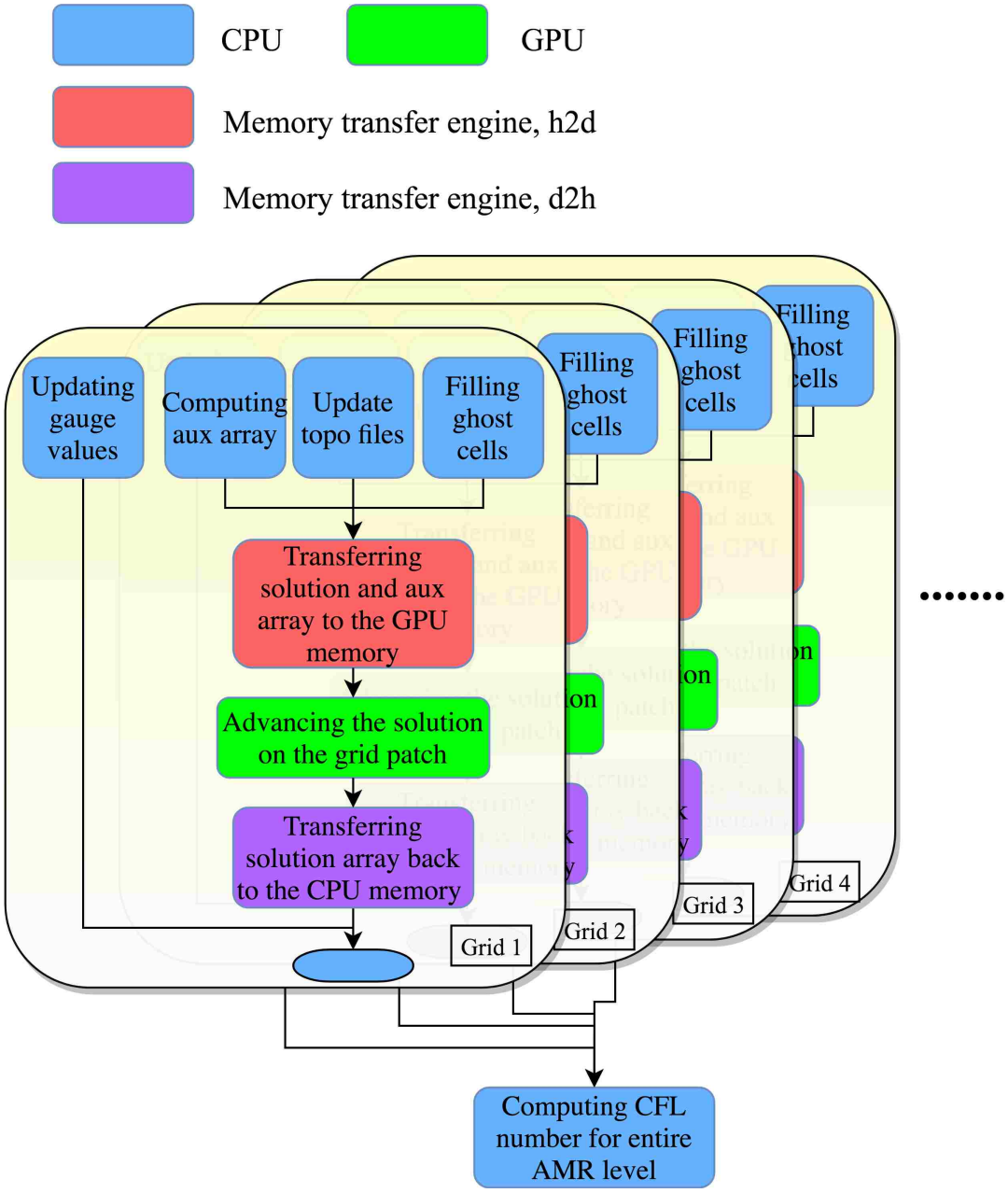


Figure 7.1: Dependency graph of some major procedures in non-AMR portion of the code. The color indicates the hardware resource a procedure requires.

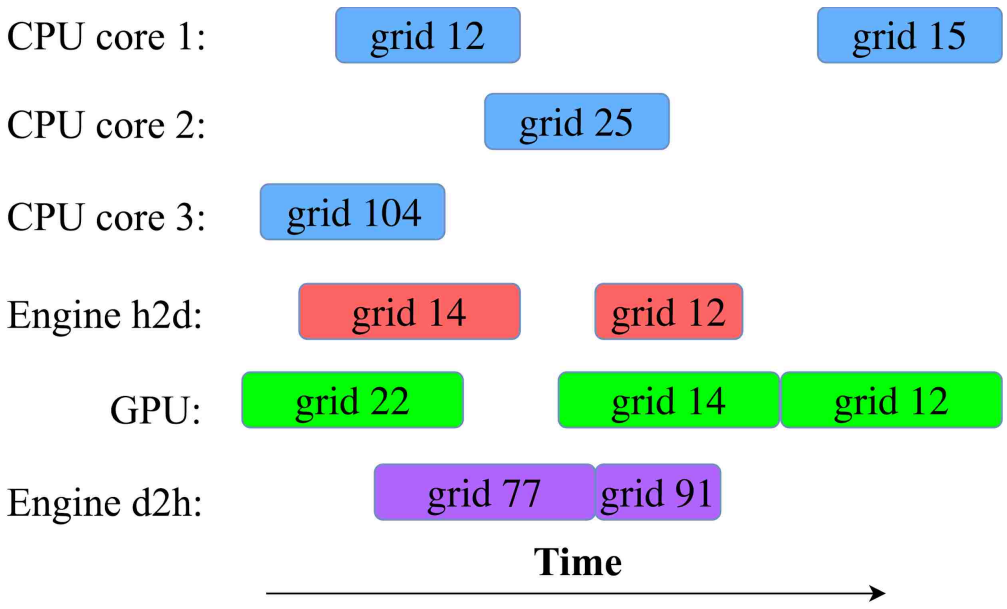


Figure 7.2: An example of different procedures in non-AMR portion of the code running concurrently along the timeline.

7.2 Memory Pool

During regridding, new grid patches are generated and old ones are destroyed. New memory must be allocated on both the CPU and the GPU for storing solution data and auxiliary data for the new grid patches, while old memory for removed grid patches must be freed. The total overhead of calling the CUDA runtime library to conduct these frequent memory operations cannot be neglected, and can even dominate execution of the code sometimes when grid patches are so small that the overhead is expensive relative to the time spent advancing the solution on grid patches. To save the cost of such frequent memory operations, a memory pool is implemented, which requests a huge chunk of memory from the system by calling the CUDA runtime library at the initial time, keeping it until the end of execution, and getting more chunks when needed. All memory allocation and deallocation requests from the code are then through this memory pool at much less cost, with no need to actually allocate/free system memory.

7.3 GPU Kernels for the Solver

7.3.1 The CUDA Programming Model

The current implementation is based on the CUDA programming model and targeted Nvidia GPUs. The architecture of Nvidia GPUs as well as explanation of the CUDA programming model are detailed in the Nvidia CUDA C programming guide [NVIDIA]. Here only a brief review is given to provide sufficient knowledge for understanding the implementation details introduced in this section.

In the CUDA programming model, each function that is written to run on the GPU is called a CUDA kernel (or GPU kernel). The code in a CUDA kernel specifies a set of instructions to be executed by multiple CUDA threads in parallel. The code can specify that some of the instructions should be executed by a certain group of threads but not the others. All threads assigned to execute a CUDA kernel are grouped into CUDA blocks. All such CUDA blocks then form a CUDA grid. CUDA blocks are independent of each other, can be sent to different Streaming Multiprocessors, and run concurrently. During the execution of a CUDA kernel, each thread is provided with in-

formation regarding which CUDA block and which thread within that block it is. Based on this information, each thread can perform its own set of instructions on a specific portion of the data.

The GPU has many different types of hardware for data storage. The three relevant types here are registers, shared memory and main memory. The registers are the fastest storage and have very low access latency but each Streaming Multiprocessor has a very limited number of registers. Each thread is assigned its own registers and thus can only access its own registers (unless special instructions are used).

The shared memory has relatively slower bandwidth and longer access latency than the registers but its bandwidth is still much faster than that of the main memory and access latency is also much shorter than that of the main memory. Each CUDA block is assigned a specified amount of shared memory, which is accessible by all CUDA threads in the CUDA block. The quantity of registers and shared memory in the Streaming Multiprocessor is limited and fixed. As a result, number of CUDA blocks that can reside in a Streaming Multiprocessor at the same time is limited by total number of registers and the amount of shared memory these CUDA blocks request. If too few CUDA blocks can reside in a Streaming Multiprocessor at the same time, the Streaming Multiprocessor has low *occupancy* and thus runs the CUDA kernel less efficiently. For this reason, it is important to minimize the number of registers used by each thread and the amount of shared memory used by each CUDA block when the CUDA kernel is designed.

The main memory is located the farthest from the chip and thus has the lowest bandwidth and the longest access latency. In principle, a CUDA kernel should have a minimal number of read and writes to main memory, especially given the fact that stencil computations for partial differential equations are often memory bandwidth bound. A simple idea in CUDA kernel design is to load all data as efficiently as possible from the main memory to the shared memory, conduct all computations using the shared memory as a buffer to avoid unnecessary accesses of main memory, and then write new data back to the main memory. However, this often causes too much shared memory usage for each CUDA block and results in very inefficient execution of the CUDA kernel.

7.3.2 Data Layout

Every 32 threads within a CUDA block are grouped as a warp, which executes the same instruction at the same time, including memory load and write operations. The hardware can execute memory request from all threads in a warp most efficiently if they access a contiguous piece of memory. This is called a coalesced access. Such characteristic of the GPU hardware makes Structures-of-Arrays (SoA) preferable over Arrays-of-Structures (AoS). With SoA layout, the same state variables, e.g. water depth h, on the entire grid patch are stored contiguously in memory, whereas with AoS format, all state variables within the same grid cell are stored contiguously in memory. Such a data layout results in strided access of the GPU memory. Namely, consecutive CUDA threads will access memory locations that are not consecutive. This can greatly reduce effective memory bandwidth since memory accesses cannot be coalesced.

The current implementation contributes to the Clawpack eco-system [Mandli et al., 2016], which uses an AoS layout since Fortran arrays are dimensioned so that $q(m, i, j)$ is the mth component (depth or momenta) in the (i, j) grid cell. However, many applications within the Clawpack eco-system will be affected if the AoS data layout is changed. Thus this work continues to use the AoS layout in the current implementation. In the first half of the dimensional splitting method, the CUDA kernel that solves the equation in the x direction reads in data in AoS but writes intermediate solution data in SoA, which is coalesced. The CUDA kernel that solves the equation in the y direction then reads in data in SoA layout in a coalesced manner and writes new solution back in AoS layout.

7.3.3 CUDA Kernel Implementations

In designing a CUDA kernel, one essential goal is to assign computational tasks to each thread. To perform time integration on gird patches with Godunov-type wave-propagation methods, the goal is to decide how to distribute to each thread the tasks of solving the Riemann problems at each cell edge, limiting waves, and updating cell values.

Figure 7.3 shows a one-dimensional slice of a grid patch along the x direction when the first

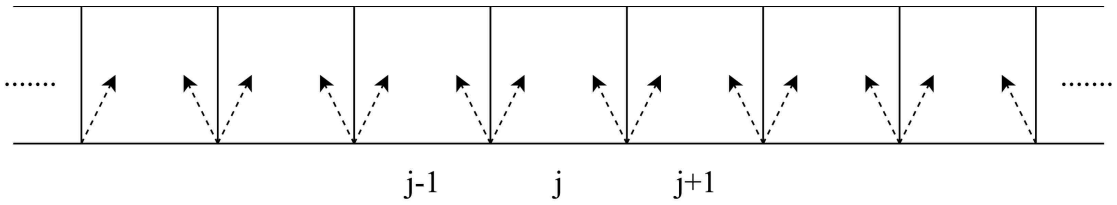


Figure 7.3: A one-dimensional slice of a grid patch along the x direction. The arrows in dashed lines represent waves from the Riemann problems at the cell edges.

step of the dimensional splitting method is conducted to get the intermediate state Q^*. Updating cell C_j depends on the two sets of waves at cell edges $x_{j-1/2}$ and $x_{j+1/2}$. When waves are limited, the waves at cell edge $x_{j-1/2}$ depends on waves at cell edges $x_{j-3/2}$ and $x_{j+1/2}$, while the waves at cell edge $x_{j+1/2}$ depends on waves at cell edges $x_{j+3/2}$ and $x_{j-1/2}$. Any of these waves depends on the two cell values around them, respectively. As a result, the solution at cell C_j depends on four neighboring sets of waves, which depend on the 5 cell values around cell C_j (including itself).

If each CUDA thread is assigned to update a cell in this one-dimensional slice, it needs to solve the four Riemann problems the cell depends on. Redundant work is performed since some Riemann problems are solved and some waves are limited by neighboring CUDA threads as well. On the other hand, if each thread is assigned to solve a Riemann problem at one cell edge in this one-dimensional slice, limit the waves, and then update the two neighboring cells with left- and right-going waves, the code must carefully avoid data racing since each cell is updated by two CUDA threads. This typically involves the usage of a synchronization mechanism called lock, which decreases the execution efficiency of the CUDA kernel.

In the current implementation, a combination of the two ideas above is implemented. Figure 7.4 shows how CUDA threads are first assigned to cell edges for solving Riemann problems and limiting waves, and then re-assigned to grid cells for updating the solution. Each solid arrow denotes assigning one CUDA thread on a cell edge. The thin arrows show the initial assignment to edges, while the thick arrows show the final assignment to cells.

In the first stage, each CUDA thread is assigned to a cell edge to solve the Riemann problem

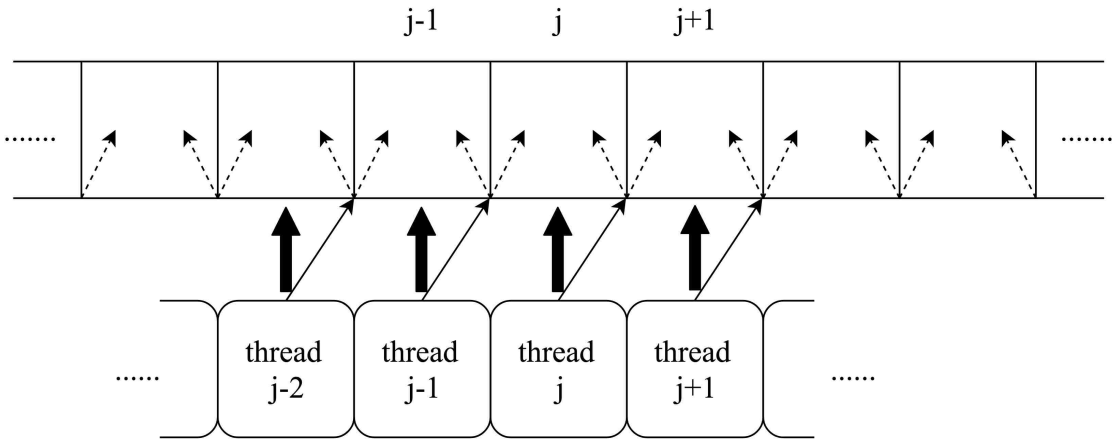


Figure 7.4: Assign CUDA threads to grid cells and cell edges.

there. The left and right state for the Riemann problem for a thread is loaded from the main memory, while resulting waves from the Riemann problem are written into the shared memory. In the second stage, each CUDA thread limits its waves to get correction fluxes at the cell edge it is assigned to. This requires reading waves from the two neighboring edges, which were produced by the two neighboring CUDA threads and stored in the shared memory. Each thread then writes the limited waves (correction fluxes) back to the shared memory.

In the last stage, each CUDA thread is assigned to update a cell (thick arrows). At this time, each thread already has left-going waves and correction fluxes at the right edge of its cell in its registers, which can be directly applied to update the cell value. The right-going waves and correction fluxes at the left edge of the same cell was produced by its left neighboring thread, and were stored in the shared memory in the last stage. Thus each thread needs to read in these waves and fluxes from the shared memory and apply them to update the cell it is assigned to. Each thread then writes the updated value Q^* back to the main memory. A similar kernel is then conducted for the second step of dimensional splitting method to get new state Q^{n+1}.

This implementation requires a kernel to read solution data from the main memory only once at the beginning of the kernel execution and write the updated solution back to the main memory

only once at the end of the kernel execution. This is done by using only a reasonable number of GPU registers for each thread and a reasonable amount of shared memory for each CUDA block. The usage of GPU registers for each CUDA thread only includes storing the state variables for left and right states of one Riemann problem, waves and wave speeds from one Riemann problem, plus any extra intermediate variables created during solving the Riemann problem, while the usage of shared memory only includes waves from one Riemann problem per thread in a CUDA block.

Chapter 8

EFFICIENT TSUNAMI SIMULATION ON ADAPTIVE GRIDS WITH THE GPUS

This section is focused on evaluating the performance of the GPU implementation. As test problems two tsunami modeling problems from recent validation studies were chosen. More details of the problems and additional comparisons of the results to those obtained using the MOST code, in simulations performed by the NOAA Center for Tsunami Research (NCTR), are available in project reports cited below. The first problem uses AMR to model waves propagating across the ocean from the 2011 Tohoku event, along with fine grid modeling around the tide gauge at Crescent City, CA. The second problem is a local tsunami from the Seattle Fault, with AMR used to focus on inundation in a coastal region very close to the fault, so the finest grid levels are activated immediately. These two problems were chosen as typical examples of different scenarios requiring AMR with potentially differing overhead.

Two machines for benchmarking the original CPU implementation and two machines for benchmarking the current GPU implementation are listed as below.

1. A single Nvidia Kepler K20x GPU with a 16-core AMD Opteron 6274 CPU running at 2.2 GHz as the host.

2. A single Nvidia TITAN X (Pascal) GPU with a 20-core Intel E5-2698 CPU running at 2.2 GHz as the host (but only 16 CPU threads are used for fair comparison with others).

3. A single 16-core AMD Opteron 6274 CPU running at 2.2 GHz.

4. A single 16-core Intel Xeon E-2650 CPU running at 2.0 GHz.

As shown in the previous sections, computational tasks are partially done on the CPU and partially done on the GPU in the GPU implementation. In the benchmarks, the CPU implementation always runs in parallel with 16 OpenMP threads, using 16 CPU cores. The CPU part of the GPU implementation are also always processed in parallel by 16 OpenMP threads, using 16 CPU cores. The GPU implementation solves the benchmark problems on machine 1 and 2 while the CPU implementation solves the same benchmark problem on machine 3 and 4. Note that machine 1 and machine 3 have the same AMD CPU while machine 2 and machine 4 have similar Intel CPU. Thus in this section, all speed-ups will be computed by comparing results on machine 1 to results on machine 3 and comparing results on machine 2 to results on machine 4. All numerical experiments in this section are conducted with double-precision floating point operations, on both the CPU and GPU.

Three metrics have been proposed that can be used to measure absolute performance of the current GPU implementation, which do not require comparing a GPU implementation to a CPU implementation. Below some quantities used in the definition of the three metrics are first defined. Along the program execution time line $t \in \mathbb{R}^+$, define $E_i^{GPU} = [t_{i,start}^{GPU}, t_{i,stop}^{GPU}]$ as the time interval that the ith GPU computation event (e.g. one of the green blocks in figure 7.2) happens. E_i^{GPU} is essentially a set of all moments that the ith GPU computation event is happening. Similarly, define E_i^{CPU}, E_i^{h2d} and E_i^{d2h} for the ith CPU computation, the ith memory transfer from the CPU to the GPU memory and the ith memory transfer from the GPU to the CPU memory, respectively. Then, all time intervals during which the GPU is doing computation, Ω^{GPU}, can be represented as $\Omega^{GPU} = \bigcup_{i=1}^{N^{GPU}} E_i^{GPU}$, where $\bigcup$ is the union operation for sets and N^{GPU} is total number of GPU computation events. Similarly, other three sets of intervals for the other three types of events are defined. All time intervals during which the CPU is doing computation, Ω^{CPU}, can be represented as $\Omega^{CPU} = \bigcup_{i=1}^{N^{CPU}} E_i^{CPU}$, where N^{CPU} is total number of CPU computation events. All time intervals during which the memory transfer from the CPU memory to the GPU memory is happening, Ω^{h2d}, can be represented as $\Omega^{h2d} = \bigcup_{i=1}^{N^{h2d}} E_i^{h2d}$, where N^{h2d} is total number of such memory transfers. All timer intervals during which the memory transfer from the GPU memory to the CPU memory is happening, Ω^{d2h}, can be represented as $\Omega^{d2h} = \bigcup_{i=1}^{N^{d2h}} E_i^{d2h}$.where N^{d2h} is

total number of such memory transfers. Lastly, define $E^{total} = [t_{start}, t_{end}]$ as the time interval that the entire program runs.

The first metric measures the proportion of time during which the GPU is doing computation, defined as

$$P_1 = \frac{|\Omega^{GPU}|}{|E^{total}|}, \tag{8.1}$$

where $|\Omega|$ represents size of the set Ω, which essentially computes the total length of all time intervals in Ω in this case. Similarly, the second metric is defined as

$$P_2 = \frac{|\Omega^{CPU}|}{|E^{total}|}, \tag{8.2}$$

which measures proportion of time during which the CPU is doing computation. The last metric measures the proportion of extra time introduced by transferring data between the CPU memory and the GPU memory

$$P_3 = \frac{|\Omega^{h2d} \bigcup \Omega^{d2h} - \Omega^{CPU} \bigcup \Omega^{GPU}|}{E^{|total|}}, \tag{8.3}$$

where $-$ is the subtract operation for sets. For two sets A and B, $A - B$ denotes all elements in A but not in B.

8.1 2011 Japan Tsunami

8.1.1 Problem Setup

The first benchmark problem is the 2011 Japan tsunami, which was triggered by an earthquake of magnitude 9.0-9.1 off the Pacific coast of Tohoku, occurred at 14:46 JST (05:46 UTC) on Friday March 11th, 2011. The earthquake source deformation files were obtained from NCTR and were converted to deformation information on uniform lat-long grids for use in our implementation. This test problem was studied as part of project funded by NCTR to validate GeoClaw for potential future use in the US Tsunami Warning Centers, and more details of this simulation along with comparisons for several other tide gauge locations and for other historical tsunamis can be found in the project report [Adams and LeVeque, 2017].

The computational domain is from longitude -240 to -100 and latitude -31 to 65 in spherical coordinates, with structured quadrilateral grid cells. Three levels of refinement are set across the ocean and around the source region (before getting close to the destination). Starting from the coarsest level (level 1) that has a resolution of 2 degrees, the refinement ratios are 5 and 6, giving a resolution of 25 minutes on level 2 and 4 minutes on level 3. A refinement tolerance parameter can be specified to guide the mesh refinement. The smaller this parameter is set, the more likely the grid will be refined to the highest level allowed in a particular region. The refinement tolerance parameter is chosen to be the wave amplitude and is set to 0.005 meter. Thus, if a region is allowed to use any of the choices above (2 degree, 24 minute, 4 minute), the region will be refined up to a maximum of level 3 when the amplitude of a wave is higher than 0.005. In addition to specifying a tolerance for flagging individual cells, regions of the domain can be specified so that all cells in the region, over some time interval also specified, will be refined to at least some level and at most some level. In the simulation, the three refinement levels mentioned above are allowed in the entire region, with other constraints in specific sub-regions. In the first 7 hours after the earthquake, a 4-minute resolution (level 3) in the region from longitude -231 to -170 and from latitude 18 to 62 is enforced. This is reverted to the choices of 2 degrees or 24 minutes after 7 hours when the wave amplitudes are below the tolerance. Then moving onward toward the destination, a 4-minute resolution in the region from longitude -170 to -120 and from latitude 18 to 62 is enforced starting at 7 hours till the end of the 13-hour simulation. A combination of the tolerance-based refinement and manually enforced refinement is used here since whether the waves are well resolved in some regions in the domain will not affect our location of interest and within our time range of interest. For instance, in Figure 8.2, the waves at bottom center (near Australia) will not affect our location of interest (point 2) until approximately two days later. As a result, enforcing those region to have relative coarse grid after a certain time saves us a huge amount of computational cost. If only the tolerance-based refinement is used, those area will end up being refined for much longer time, and to much finer level. The choice of such refined region are based on practical tsunami modeling experience. A new variant of the code is under development that uses the solution of an adjoint problem to better guide the adaptive refinement and automatically determine what waves in the

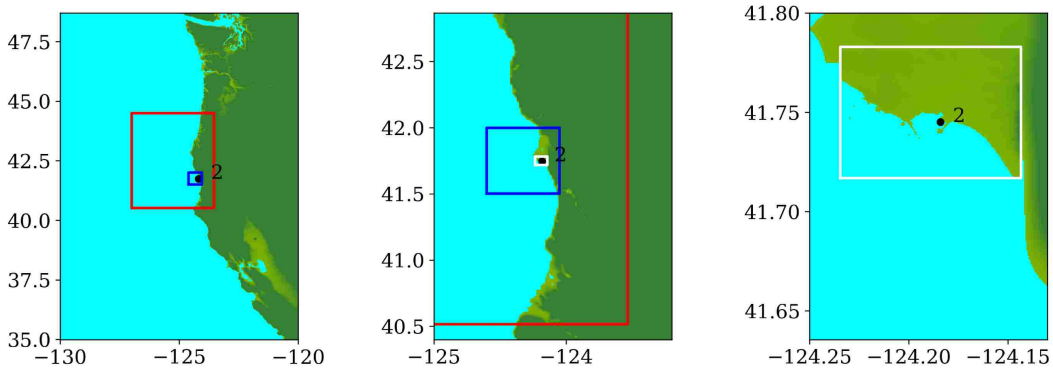


Figure 8.1: Three refinement regions around Crescent city with higher resolution and location of gauge 2. Red: level 4, 1-minute resolution; Blue: level 5, 12-second resolution; White: level 6, 2-second resolution.

forward solution at any regridding time can potentially affect the solution in the target region of interest [Davis and LeVeque, 2016].

Near Crescent City, the destination of interest, three higher higher levels of refinement regions are enforced to resolve for smaller-scale flow features near the coast:

1. Level 4 with 1-minute resolution is enforced starting from 8 hours after the earthquake, in the region from longitude -126.995 to -123.535 and from latitude 40.515 to 44.495.

2. Level 5 with 12-second resolution is enforced starting from 8 hours after the earthquake, in the region from longitude -124.6 to -124.05 and from latitude 41.502 to 41.998.

3. Level 6 with 2-second resolution is enforced starting from 8.5 hours after the earthquake, in the region from longitude -124.234 to -124.143 and from latitude 41.717 to 41.783.

Figure 8.1 shows the three refinement regions as well as location of gauge 2, where the time series of water surface elevation is recorded.

To ensure solution data on an entire grid patch can fit into the cache of the CPU for data locality, the size of each grid patch is limited to 128 by 128 for both the GPU and CPU cases. The Godunov-type dimensional splitting scheme is used with second-order MC limiter applied to the waves. The problem is simulated for a simulation time of 13 hours.

8.1.2 Simulation Results

Figure 8.2 and Figure 8.3 show snapshots during the simulation for the entire computational domain and near Crescent city, colored by $\zeta(x, y, t)$ defined as

$$\zeta(x,y,t) = \begin{cases} h(x,y,t), & \text{if } B(x,y) > 0 \text{ (the flow depth),} \\ h(x,y,t) + B(x,y), & \text{if } B(x,y) \leq 0 \ (\eta\text{, water surface elevation).} \end{cases} \tag{8.4}$$

During the tsunami, wave height data were recorded at 4 DART buoys (Deep-ocean Assessment and Reporting of Tsunamis) near the earthquake source, the locations of which are shown in Figure 8.4. The blue rectangle in the figure indicates the extent of the earthquake source. However, most of the sea floor deformation is inside the red rectangular region, where one-minute topography files are used to make sure the region is well resolved. The wave heights predicted by the current GPU implementation at the 4 DART buoys are shown in figure 8.5 and compared against observed data. The comparison shows the predicted results agree quite well with observed data at the 4 DART buoys.

Recall that the current implementation uses a dimensional splitting scheme with no refluxing. To show that this simplification gives comparable results to the original GeoClaw code, Figure 8.6 gives time series of surface elevation recorded at a tide gauge near Crescent City, California, United States, during the 2011 Japan Tsunami. The observation has been detided by subtracting the predicted tide level from it to remove the influence of tide level. Sample result from another well-known tsunami model MOST [Titov and Gonzalez, 1997] have also been included for comparison. The comparison shows that a simplified GeoClaw CPU code that implements such a dimensional splitting scheme with no refluxing gives very close results to those produced by the original GeoClaw and agree well with another model and observed data. The current GPU implementation

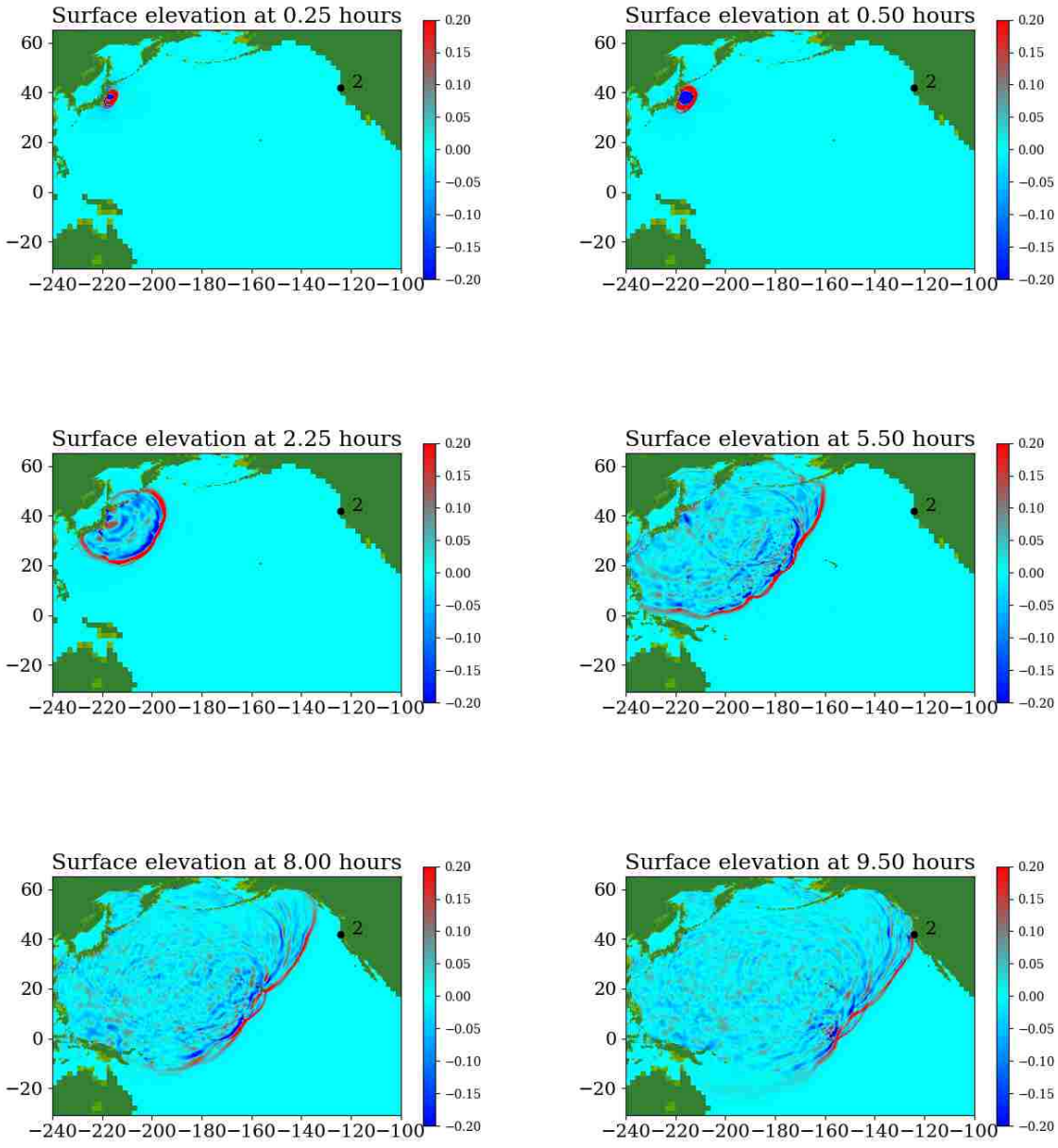


Figure 8.2: $\zeta(x, y, t)$ at 5.5 hours and 9.5 hours after the Japan 2011 earthquake.

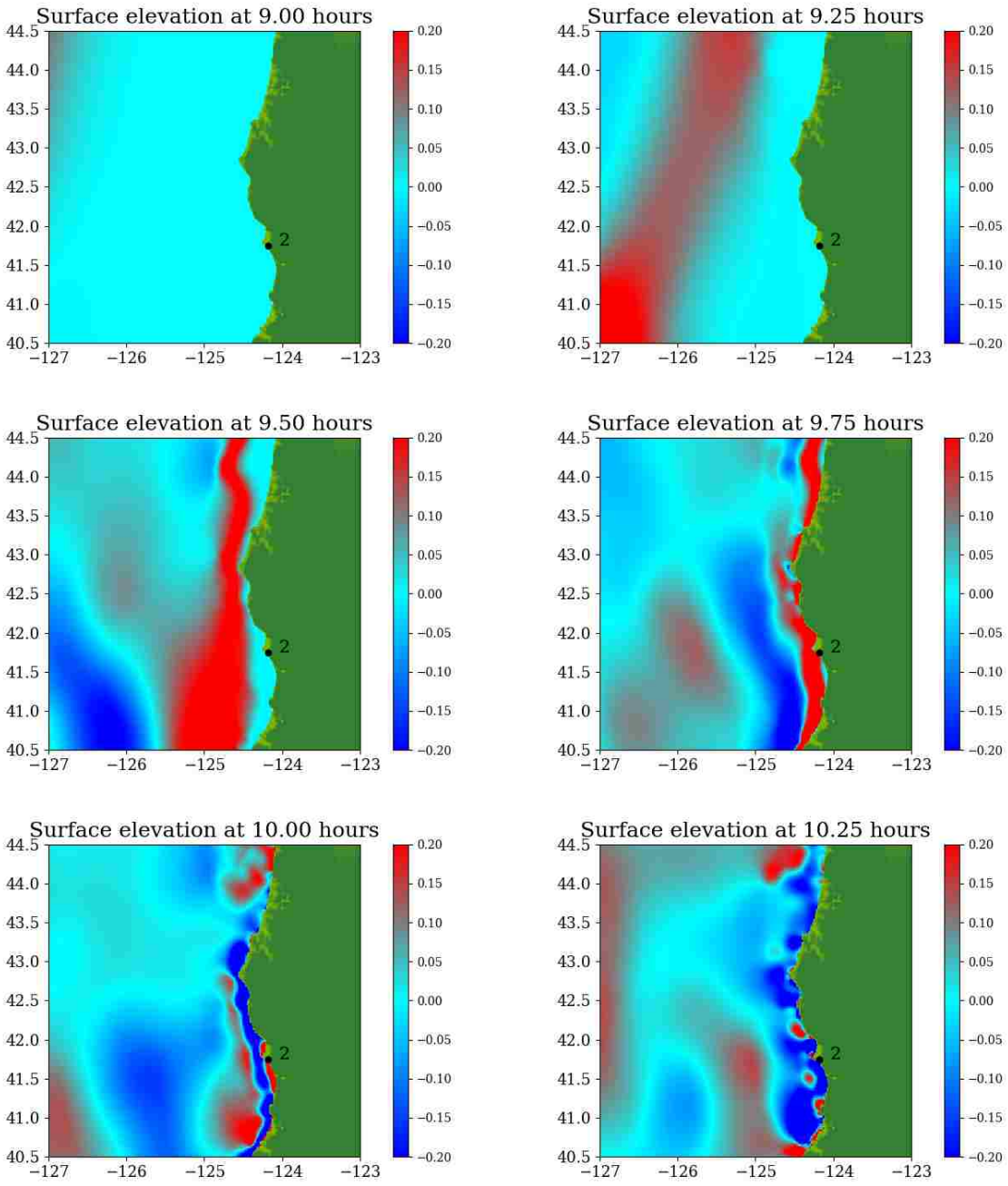


Figure 8.3: $\zeta(x, y, t)$ at 9.25 hours and 9.75 hours after the Japan 2011 earthquake, zoomed in near Crescent city.

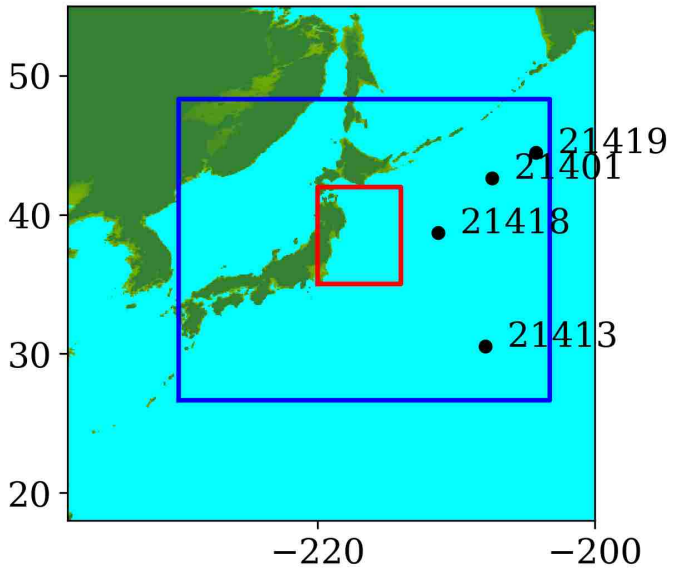


Figure 8.4: Japan 2011 earthquake source and DART buoys locations. The coordinates for each DART buoys are: 1) gauge 21401, longitude -207.417, latitude 42.617; 2) gauge 21413, longitude -207.883, latitude 30.515; 3) gauge 21418, longitude -211.306, latitude 38.711; 4) gauge 21419, longitude -204.264, latitude 44.455.

gives identical results to this simplified version of the original GeoClaw and is not shown in the Figure. Table 8.1 shows the total number of cell updates and the total number of the time steps taken on each AMR level for the simulation of the Japan 2011 tsunami. Note that a big portion of the computational cost is spent on level 3 grid patches, which cover a large area in the middle of the Pacific ocean. Although level 3 grid patches are advanced for only approximately 5000 time steps, the total number of cell updates on this level is still larger than on level 6 grid patches, which are advanced almost 4 times more often but cover a much smaller region and have fewer cells. One limitation of the current implementation is that the total number of cells allowed on each AMR level is limited by the amount of the GPU memory, which is typically smaller than that of the host CPU machine. The refinement ratio and criteria in both numerical experiments in this chapter have

Table 8.1: Total number of cell updates and total number of the time steps taken on each AMR level for the simulation of Japan 2011 tsunami.

AMR level	Total number of cell updates	Total number of the time steps taken
1	5.44E+05	162
2	4.44E+07	877
3	6.94E+09	5024
4	3.52E+08	5330
5	4.17E+08	10472
6	6.92E+08	18544
Total	8.45E+09	40409

been carefully chosen such that number of cells on each level never goes beyond the limit of GPU memory during the simulation.

Figure 8.7 shows total running time and proportion of the three components on 4 machines. The total speed-ups are 4.3 on machine 1 and 6.4 on machine 2 for the current GPU implementation. Note that since time spent on the non-AMR portion decreases on machine 1 and 2, the cost for regridding and updating take up larger portion of the total run time. However, one could still only gain a very limited additional performance increase if the regridding and updating processes were implemented on the GPU. Amdahl's law states theoretical speed-up of the execution of a whole program is

$$S(s) = \frac{1}{(1-p) + \frac{p}{s}}, \tag{8.5}$$

where S is theoretical speed-up of the execution of the whole program, s is the speed-up of the portion that is accelerated, p is proportion of total running time that the accelerated portion takes. Further more, one has $S(s) \leq \frac{1}{1-p}$, where the equality is achieved when s approaches ∞ in equation (8.5). From Amdahl's law, even if the regridding and updating processes are implemented on the GPU and are accelerated infinitely, the entire program only get roughly 1.2 speed-up on

Table 8.2: The three metrics measured from simulating the Japan 2011 tsunami on machine 1 and machine 2.

	machine 1	machine 2
P_1	46.92%	64.20%
P_2	84.50%	79.30%
P_3	3.98%	2.90%

machine 1 and 2.

Table 8.2 shows the three metrics for the current GPU implementation running on machine 1 and machine 2 when the Japan 2011 tsunami is simulated. The proportion of GPU computation (P_1) reaches about 50% on machine 1 and a higher percentage of 64% on machine 2. This could be due to the fact that machine 2 has a newer GPU which has much lower overhead for kernel launch and memory transfer. The proportion of CPU computation (P_2) are around 80% for both machines. In other words, during 20% of the total running time, the CPU is idle. P_3, the extra time introduced by transferring data between the CPU and the GPU memory, is less than 5% for both machines. This shows that even if the data can be transferred infinitely fast between the CPU and the GPU memory so the data transfer has no effect on execution time at all, the total running time of the entire program can be reduced by at most 5%. Thus having to transfer data between the CPU and the GPU memory is not a critical issue that affects the performance of the code.

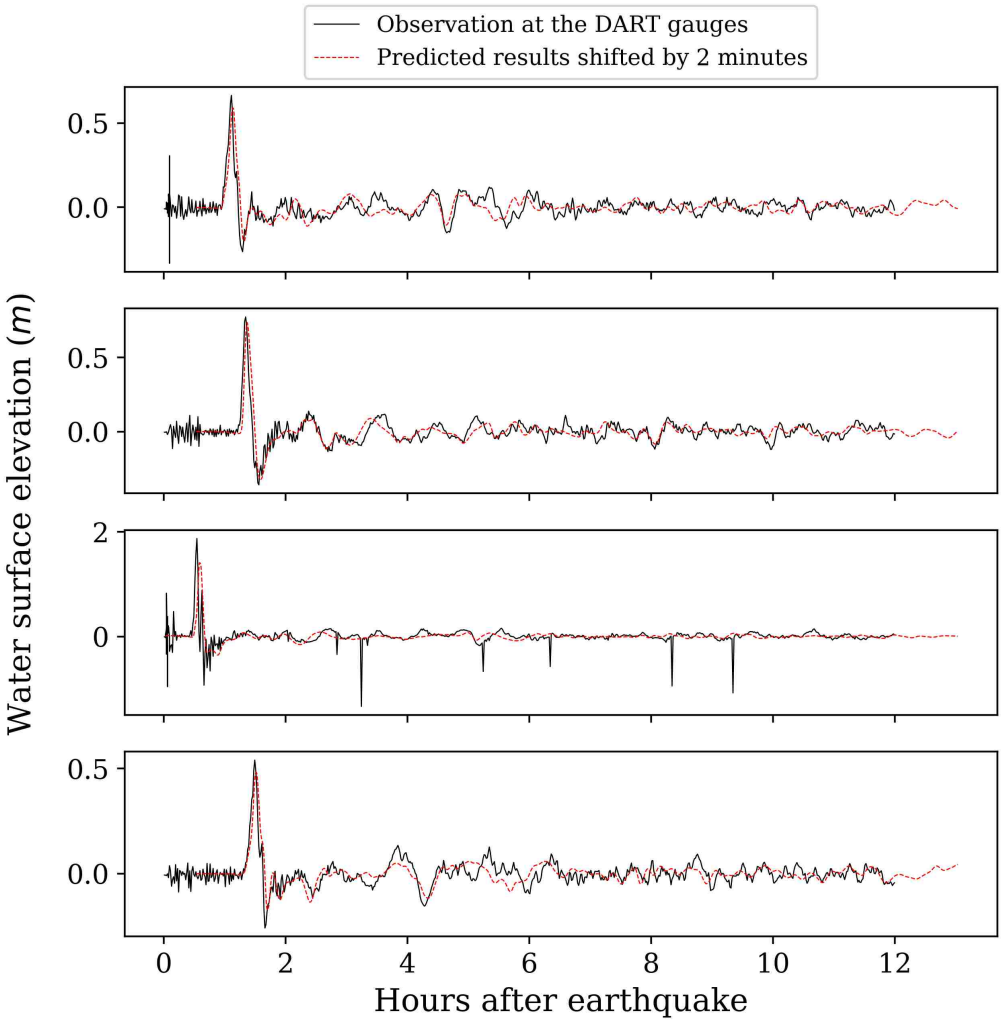


Figure 8.5: Water surface elevation at 4 DART buoys. From top to bottom: gauge 21401, gauge 21413, gauge 21418, gauge 21419.

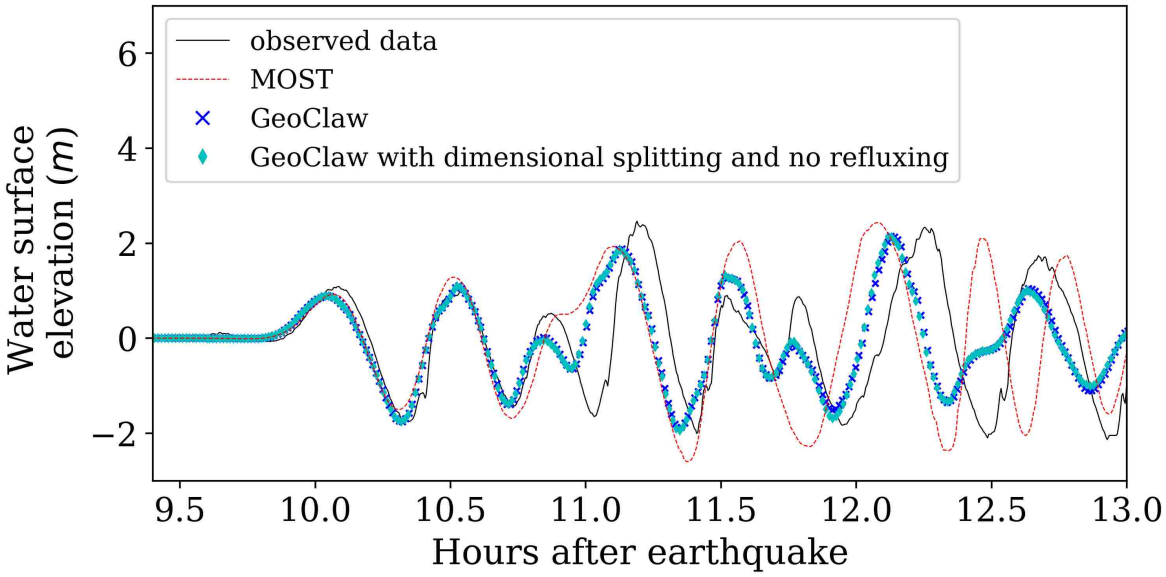


Figure 8.6: Water surface elevation at gauge 2 (location: longitude -124.1840, latitude 41.7451) near Crescent city. Time series from the MOST model are shifted by 6 minutes. All other time series from numerical results are shifted by 6.5 minutes.

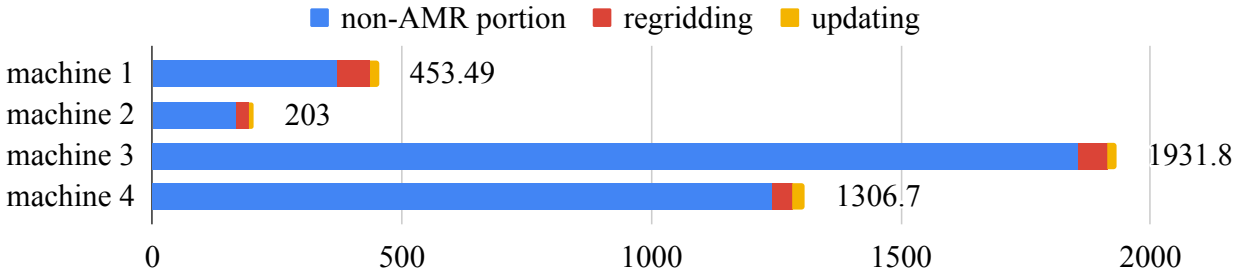


Figure 8.7: Wall time (in seconds) of entire program on simulating the Japan 2011 tsunami, for original CPU implementation running on machine 3 and machine 4, and the current GPU implementation running on machine 1 and machine 2.

8.2 A Local Tsunami Triggered by Near-Field Sources

8.2.1 Problem Setup

The second benchmark problem is the modeling of a local tsunami that is triggered by a near-field earthquake, which typically hits the shoreline much earlier than a tsunami triggered by a far-field earthquake. The tsunami is triggered by a hypothetical Mw 7.3 earthquake on the Seattle Fault, which cuts across Puget Sound (through Seattle and Bainbridge Island, see figure 8.8) and can create a tsunami that can cause significant inundation and high currents in some coastal communities around the Puget Sound. The event was designed to model an earthquake that occurred roughly 1100 years ago, and for which geologic data is available for the uplift or subsidence at several locations. Here, the focus is on modeling this local tsunami and predicting its impact on Eagle Harbor at the Bainbridge island, the location of which is shown below in figure 8.9. The ground deformation file for generating the tsunami was obtained from the NCTR, and this test problem has been used for recent model comparison and validation study of GeoClaw and MOST as part of a tsunami hazard assessment of Bainbridge Island. More details about the modeling, along with additional comparisons of model results from the two codes, can be found in the project report [Titov et al., 2018].

Figure 8.9 also shows the computational domain, which is from longitude -123.61 to -122.16 and latitude 47 to 48.7 in spherical coordinates, with structured quadrilateral grid cells. Since this is a local tsunami in an enclosed Sound surrounded by land, the tsunami waves soon get reflected by shorelines and spread out to cover the full domain very soon after the earthquake. Thus, instead of using a refinement tolerance parameter, mesh refinement is enforced everywhere in the domain regardless of wave amplitude, and never regenerate new grid patches. Four levels of refinement are used, as denoted by the rectangles in figure 8.9 that denote regions where refinement is enforced. Starting from the coarsest level (level 1), which has a resolution of 30 minutes, the refinement ratios are 5, 3 and 6, giving a resolution of 6 minutes on level 2, 2 minutes on level 3 and $\frac{1}{3}$ minutes on level 4. Note that for this benchmark problem, a large proportion of the domain is dry land and the shorelines are relatively much longer and more complex. As a result, many branches occur along

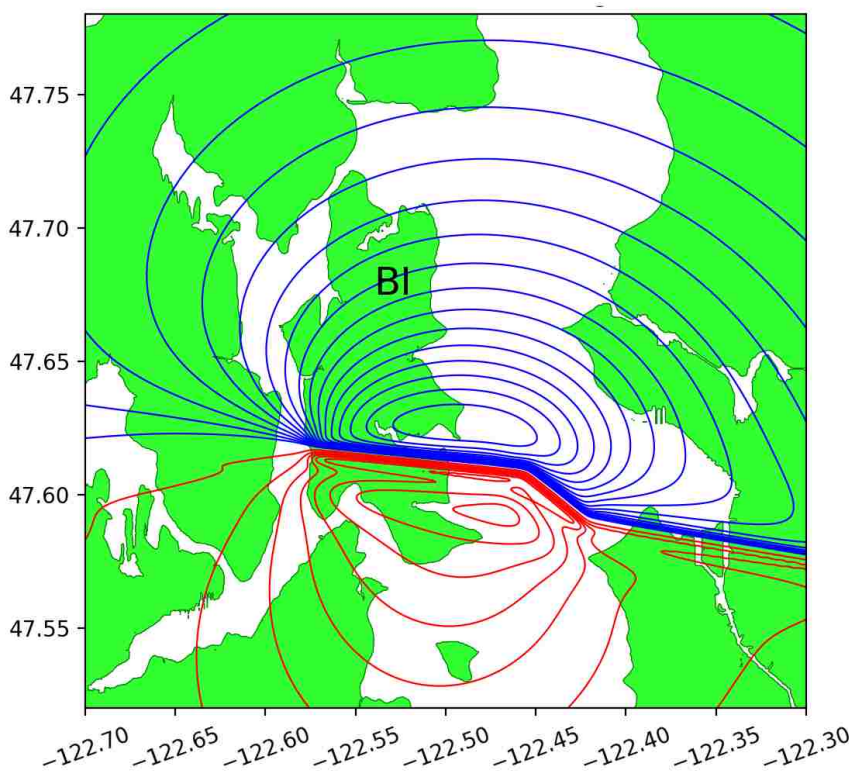


Figure 8.8: Surface displacement for the hypothetical Seattle Fault earthquake, with Bainbridge Island labelled BI. Eagle Harbor is just north of the fault on the east side of the island. Red contours show uplift at levels $0.5,\ 1,\ 1.5,\ \ldots$ meters, blue contours show subsidence at levels $-0.05, -0.1,\ \ldots$ meters.

the execution path of solving Riemann problems of the shallow water system since more different situations arise, e.g. a Riemann problem with one state being dry initially but becoming wet, or staying dry, depending on the flow depth and velocity in the neighboring cell. For the GPU, if the 32 threads in a warp do not take the same execution path, each extra branch will be executed by the entire warp, introducing significant extra execution time. Thus the irregularity of water area in this benchmark problem is challenging for some CUDA kernels to use the GPU hardware efficiently .

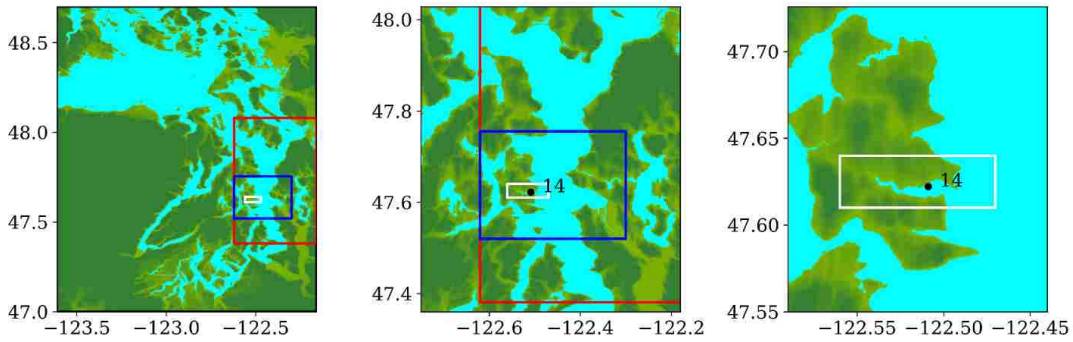


Figure 8.9: Computational domain and refinement regions for tsunami inundation triggered by the Seattle fault. Red rectangle shows the region where level 2 refinement is enforced. Blue rectangle shows the region where level 3 refinement is enforced. White rectangle shows the region where level 4 refinement is enforced.

8.2.2 Simulation Results

Figures 8.10 and 8.11 show snapshots from the simulation at several moments during the simulation in the Puget Sound and near Eagle Harbor, colored by $\zeta(x, y, t)$ defined in equation (8.4). The black solid line denotes the original shoreline before the earthquake. At the entry of the harbor, deep inundation occurred at several places as early as only 3 minutes after the earthquake. Even at the very end of the Eagle Harbor, the influence from the tsunami is also significant, causing more than 2-meter deep inundation in several places starting at 9 minutes after the earthquake. One wave gauge is placed inside Eagle Harbor to record the inundation depth during the tsunami. Figure 8.9 shows the location of the wave gauge. As this is a hypothetical event for modeling an earthquake roughly 1100 year ago, there is no surface elevation observation available for comparison. Hence, the results from the current implementation are compared with those from the MOST tsunami model [Titov and Gonzalez, 1997]. Additional comparisons of GeoClaw and MOST results can be found in the comparison study recently performed by Titov et al. [2018]. For that study the original CPU version of GeoClaw was used, with the un-splitting algorithm and refluxing, but it has been

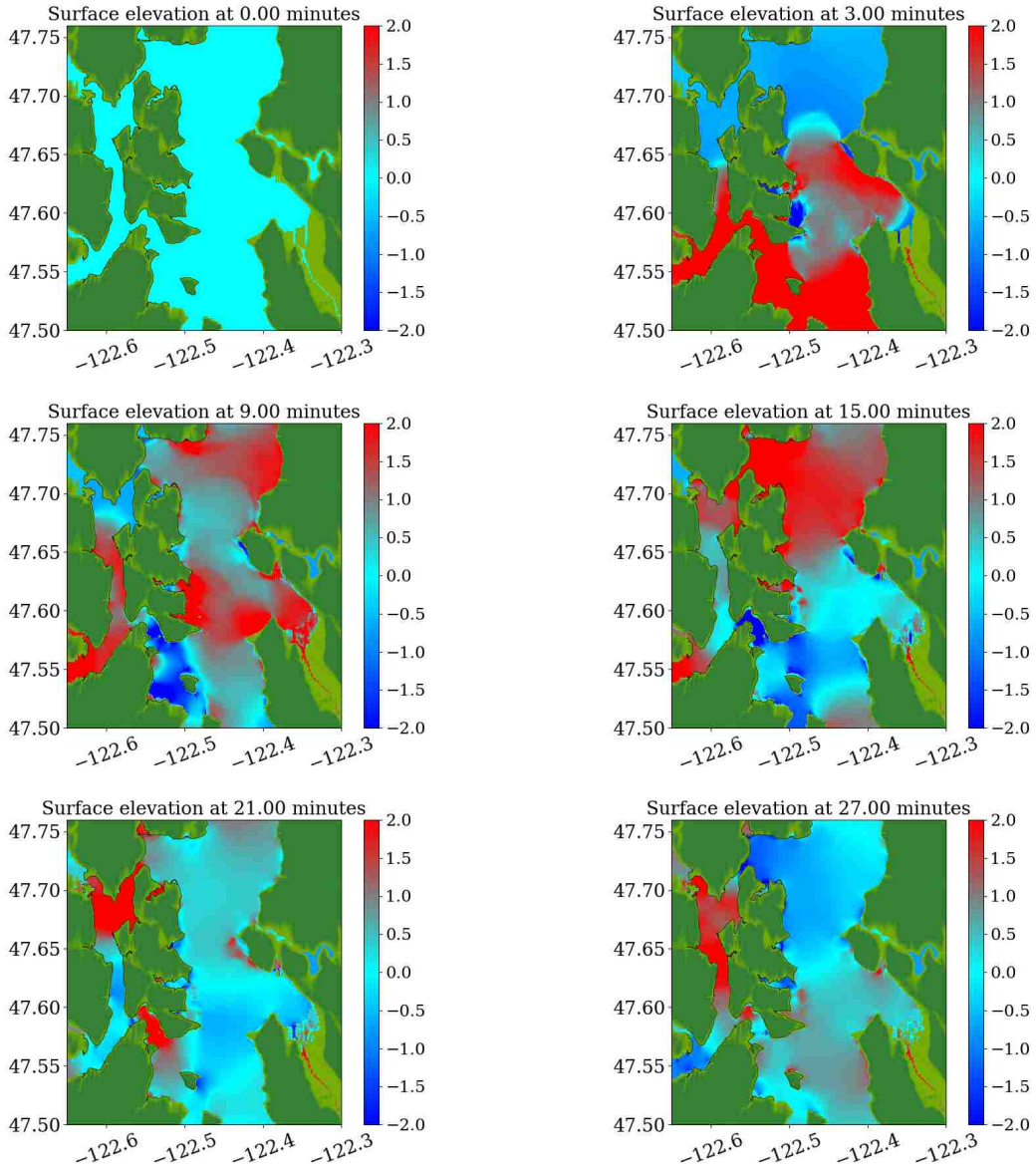


Figure 8.10: $\zeta(x, y, t)$ in Puget Sound after a tsunami triggered by Seattle fault rupture.

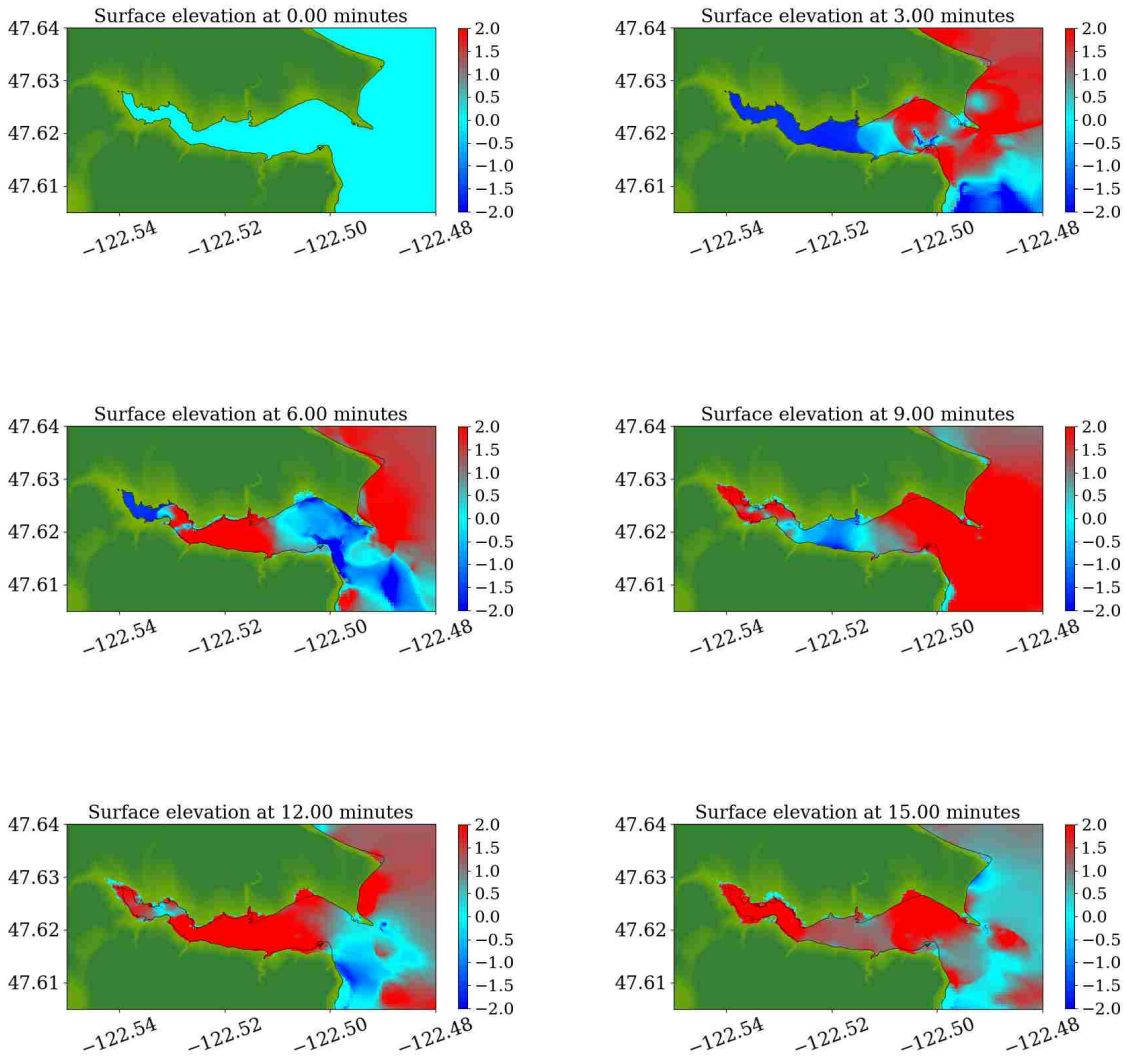


Figure 8.11: $\zeta(x, y, t)$ in Eagle Harbor of Bainbridge island after a tsunami triggered by Seattle fault rupture. The solid line denotes location of the shoreline at initial.

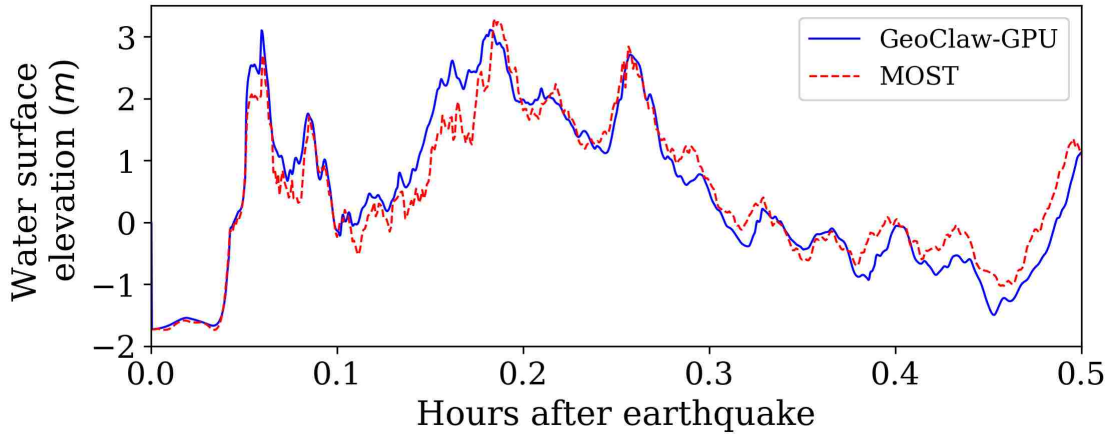


Figure 8.12: Water surface elevation at a gauge (location: longitude -122.5089, latitude 47.6222) inside Eagle Harbor of Bainbridge island.

confirmed that very similar results are obtained with the GPU code, at least in Eagle Harbor. Table 8.3 shows the total number of cell updates and total number of the time steps taken on each AMR level for the simulation of a tsunami triggered by the Seattle fault rupture. The majority (88%) of the computational cost is spent on level 4, which is designed to only cover the small area of interest, around Eagle Harbor on Bainbridge island.

Figure 8.13 shows total running time and proportion of the two components on 4 machines (no regridding process since it is never conducted) The total speed-ups are 3.7 on machine 1 and 5.0 on machine 2 for this benchmark problem. For the original CPU implementation, the non-AMR portion takes 98% and 99% of the total computational time, which indicates high potential of benefiting from optimizing the performance of this portion. Although the proportion of the non-AMR portion increases for the current implementation on machine 1 and machine 2, it still takes more than 95% of the total computational time, showing great potential for further improvement.

Table 8.4 shows the three metrics for the current GPU implementation running on machine 1 and machine 2 when the Seattle Fault tsunami is simulated. Similar values are obtained for all three metrics, showing consistency and validity of the three metrics on evaluating GPU implementation

Table 8.3: Total number of cell updates and total number of the time steps taken on each AMR level for the simulation of a tsunami triggered by Seattle fault rupture.

AMR level	Total number of cell updates	Total number of the time steps taken
1	8.41E+06	237
2	5.28E+08	1165
3	9.03E+08	3484
4	7.40E+09	20862
Total	8.84E+09	25748

Table 8.4: The three metrics measured from execution of the code on machine 1 and machine 2, simulating the Seattle Fault tsunami.

	machine 1	machine 2
P_1	60.76%	57.39%
P_2	84.80%	84.60%
P_3	6.87%	1.77%

with different tsunami problems.

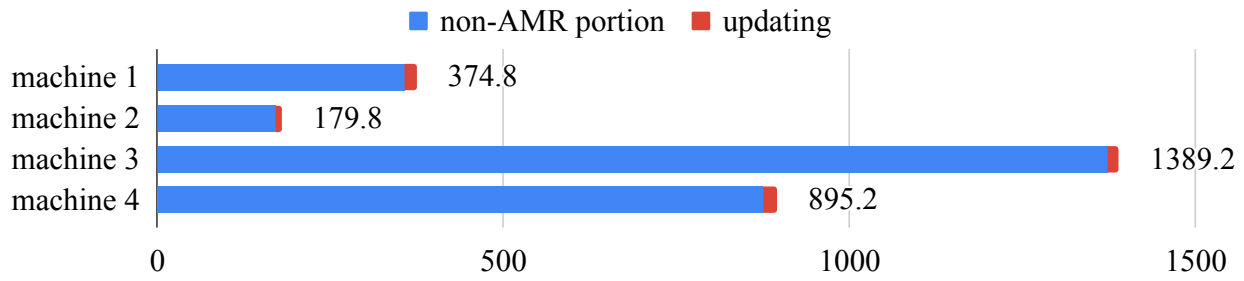


Figure 8.13: Wall time (in seconds) of entire program on simulating the Seattle Fault tsunami, for original CPU implementation running on machine 3 and machine 4, and the current GPU implementation running on machine 1 and machine 2.

8.3 Conclusions

The shocking fatalities and infrastructure damage caused by tsunamis in the past two decades highlight the importance of developing fast and accurate tsunami models for both forecasting and hazard assessment. This chapter presents the development of a fast and accurate GPU-based version of the GeoClaw code using patched-based AMR. Arbitrary levels of refinement and refinement ratios between levels are supported. The surface elevation at DART buoys and wave gauges in the benchmark problems show the ability of the current tsunami model to produce accurate results in tsunami modeling. With the GPU, the entire tsunami model runs 3.6–6.4 times faster than an original CPU-based tsunami model for several benchmark problems on different machines. As a result, the Japan 2011 Tohoku tsunami can be fully simulated for 13 hours in under 3.5 minutes wall-clock time, using a single Nvidia TITAN X GPU. Three metrics for measuring the absolute performance of a GPU-based model are also proposed to evaluate the performance of the current GPU implementation without comparing to others, which shows the ability of the current model to efficiently utilize GPU hardware resources. Other hazards such as storm surge (e.g. Mandli and Dawson [2014]) and dam failures (e.g. George [2011]) can also be modeled with GeoClaw and can also benefit from this GPU-accelerated version of GeoClaw.

Chapter 9

SUMMARY AND FUTURE WORK

9.1 Summary

The primary focus of this work was on developing accurate and fast numerical models for tsunami modeling. The first part described the development of a two-dimensional (2D) and a three-dimensional (3D) tsunami inundation models and the comparison between them. The GeoClaw software used to build the 2D numerical model was then accelerated with GPUs in the second part of the dissertation. Both works contributed to the study and improvement of tsunami models.

In particular, part I of the dissertation built a 3D and a 2D tsunami inundation models of a 1:50 model-scale representative of the town of Seaside, Oregon in the United States, to simulate the challenging inundation phase of tsunamis. The 3D model was based on solving the 3D RANS equations with OpenFOAM, while the 2D model used the GeoClaw software to solve the shallow water equations. The constructed environments (buildings and seawalls) were explicitly modeled, which added complexity and challenge to the problem. The 3D model could provide much more details of the flow and direct prediction of tsunami loads on structures of interest but at a much higher computational cost compared to the 2D model. Due to the limitation of computational resources, only sub-sections of the entire wave basin could be modeled in a single simulation by the 3D model. Numerical experiments were thus conducted to find proper selection of sub-sections for areas of interest. The study found that, with proper selection of the sub-sections, the 3D model could model the entire domain without loss of accuracy, which greatly alleviated the expensive computational demand required by the 3D model.

Water depth, velocities in two horizontal directions and momentum flux from the two numerical models were compared with measurement in the experiment and they agreed reasonably well in most time periods, except for the time near the initial impact of the wave on the coastal structures.

Near the initial-impact time region, the 2D GeoClaw model had more difficulty in predicting the flow parameters due to transient characteristic of the flow. Although the 3D OpenFOAM model could do better in capturing these turbulent flows, it did so at an expense of much more computational resources. The 2D GeoClaw model required much less computational resources and could model the entire basin in a single simulation. By carefully comparing velocity field in the numerical results with the experimental measurement, a potential problem was found in the optical approach used to measure velocity in the experiment. The optical approach assumed the velocity of the leading edge of the bore was the maximum velocity in the flow, while the numerical results indicated this was not always true and maximum velocity could occur somewhere after the bore front. Thus using the optical approach to estimate peak velocity could underestimate it. Because the tsunami load is in proportion to the square of velocity, the error in the estimation of peak velocity could result in large error in the estimation of tsunami load.

Tsunami inundation models are very useful for the planning of evaluation route and for the design and construction of buildings and structures in tsunami inundation zones. While prediction of the flooding depth and flow speed around these buildings could provide very useful information for these goals, prediction of fluid forces on buildings is also very useful, or even necessary for designing and constructing critical structures such as evacuation structures, hospitals or fire stations, which are expected to maintain certain levels of capability in a tsunami event. Part I of the dissertation also compared the capability of the 3D and 2D models to predict tsunami load on structures. Although the 3D model was much more expensive than the 2D model, it could produce direct prediction of tsunami loads on the structures by integrating the pressure and shear stress on their surfaces. The 2D model could only do so indirectly by extrapolating tsunami forces from flow quantities with empirical approaches. In this study, the definition of the drag coefficient was used to extrapolate fluid forces from water depth and flow velocity. It was shown that the inclusion of constructed environments in the numerical model was important for getting a reasonably good estimation of the fluid forces, although a better choice of the drag coefficient other than the typical choice of 2.0 might be more proper. Overall, it could be hard to get accurate prediction of fluid forces for all structures and at all time during the simulation with this approach since it was found

that the best choice of the drag coefficient not only varied with time but also highly depended on location and the surrounding environment which could affect the direction of the flow.

In addition to providing tsunami forces on a structure as a whole, the 3D model could also predict flow forces on components of a structure. Forces on individual walls over time output by the 3D model were discussed. It was found that the walls of a building could be impacted by the flow at different times and in different directions, which indicated that even when the net force on a building was not very large, forces on individual components such as walls could be quite large and result in critical local damage.

In part II of this dissertation, the GeoClaw software used to build the 2D model in part I was accelerated with the Graphics Processing Units (GPUs) by using the CUDA programming model. The finite volume methods, governing equations and the adaptive mesh refinement (AMR) algorithm in GeoClaw were first described to help explain the design and implementation of the GPU code. The use of Adaptive Mesh Refinement (AMR) in the code is necessary for alleviating computational cost in modeling transoceanic tsunamis and adds challenges to the implementation, including dynamic memory structure creation and manipulation, balanced distribution of computing loads between the CPU and the GPU, and optimizations to minimize global memory access and maximize arithmetic efficiency in the GPU kernel.

In particular, a hybrid CPU/GPU approach was used to utilize the computational resources from both the CPU and the GPU. A custom memory pool was developed to reduce the overhead of frequent allocation and deallocation of CPU and GPU memories. To exploit the concurrency between different procedures in the code, a dependency graph for multiple procedures in the GeoClaw software was constructed. Different procedures were allowed to run concurrently as long as the type of hardware a procedure required is available and it satisfied such dependencies, which were enforced through a combination of rearrangement of CPU procedures and GPU kernel launches, the use of OpenMP directives and CUDA streams, and proper synchronizations between CPU threads and between the CPU and the GPU. Several CUDA kernels were also developed for integrating the solution on computational grids and adjusting global time step sizes. To achieve a balance between redundant computation and inefficient memory access, dimensional splitting method was

implemented to integrate the solution. Within a time step, each CUDA thread was first assigned to solve a Riemann problem at a cell edge and then assigned to a cell to update the solution using the waves from its neighboring cell edges.

Two realistic tsunami modeling problems from recent validation studies, the Japan 2011 tsunami and a local hypothetic tsunami caused by the Seattle fault, were then used as benchmarks to evaluate performance of the current GPU code. Wave heights at several wave gauges predicted by the current GPU code were compared against those from other well validated codes and from real observations and they agreed quite well. For the Japan 2011 tsunami, six levels of refinement was used to obtain a very wide range of resolution for the grids (from $O(10^5)$ m in the middle of the ocean to $O(10^1)$ m inside a harbor where wave heights were measured). By combining the AMR algorithm and GPU acceleration, the GPU code was able to simulate the Japan 2011 tsunami for 13 hours in under 3.5 minutes wall-clock time, using a single Nvidia TITAN X GPU. Performance of the GPU-accelerated GeoClaw was evaluated in two ways. The first approach compared the speed of the GPU code against that of the original CPU code, which found that the GPU code could run 3.6-6.4 times faster on two types of GPUs that were used than the original GeoClaw running on CPUs for the two benchmark problems. The other approach evaluated performance of the GPU code by using three metrics that were proposed to reflect the absolute performance of the GPU code without having to comparing it against other code, which showed efficient usage of hardware resources by the GPU code.

9.2 Future Work

This dissertation contributed to the study of tsunami models that solve a set of partial differential equations (PDEs) that govern the underlying physics. These tsunami models could improve our capability of responding to future tsunami hazards and the safety and resilience of coastal communities. These PDEs-based models, however, have their own limitations, such as heavy computational cost and possibly not being the most proper description of some underlying physics. Recently, machine learning techniques have been shown to succeed in many areas such as natural language processing and computer vision. Modern machine learning techniques have rarely been

applied in the tsunami research community and have great potential to overcome some limitations of traditional PDEs-based approach.

One promising application of the machine learning approach to tsunami hazards study is tsunami forecasting for early warning. The inversion process that gives the input to a tsunami model in the current early warning system often takes hours [Wei et al., 2008] and is based on limited data. Thus the current early warning system for the Pacific Northwest only works well for far-field tsunamis, such as those originating in Japan, Alaska, or Chile for example. For tsunamis originating in the near field, in particular from thrust earthquakes on the CSZ, this warning system will be of limited use, given the fact that the tsunami from such an earthquake will reach many coastal locations in less than 30 minutes and is sensitive to the spatial and temporal distribution of seafloor motion [Melgar et al., 2016, LeVeque et al., 2018]. Another type of tsunami that the current operational tsunami early warning system is not designed for is a non-seismically generated tsunami such as a tsunami triggered by landslide or volcanic eruption. Seismic inversion cannot be used to infer tsunami initial condition since the tsunami is not triggered by an earthquake. Such a tsunami may affect only nearby areas, but can be devastating at those locations since they often reach the coast in a very short time.

A machine-learning-based tsunami model can address such limitations in the current tsunami warning system. This gives us a direct warning approach that does not rely on the time-consuming inversion for tsunami initial condition and the running of forward tsunami models after that. Such a direct warning approach measures information and infers expected coastal hazard severity directly from the measurement. One example of direct warning approaches is using current speeds measured by high-frequency radar and surface elevation measured by moored GNSS (Global Navigation Satellite System) buoys near-shore to infer tsunami hazards along the nearby coast.

For future work, a machine learning model can be trained with observation of some signals from nearby coast, such as current speeds and wave height, as input, and tsunami hazard severity along the coast as output. Then the model can be deployed to forecast tsunami hazards along the coast by monitoring relevant signals in real time. The training samples can be generated from well-validated PDEs-based models like GeoClaw and/or real observation. Such an approach not

only can detect non-seismically-generated and near-field tsunamis, but also runs much faster and thus gives earlier warning and more response time to the coastal communities.

BIBLIOGRAPHY

American Society of Civil Engineers (ASCE). Minimum Design Loads for Buildings and Other Structures, Standard ASCE/SEI 7-10, 2013.

American Society of Civil Engineers (ASCE). Minimum Design Loads for Buildings and Other Structures, Standard ASCE/SEI 7-16, 2016.

M. Acuña and T. Aoki. Real-time tsunami simulation on multi-node GPU cluster. In *ACM/IEEE conference on supercomputing*, 2009.

L. M. Adams and R. J. LeVeque. Geoclaw model tsunamis compared to tide gauge results final report. Technical report, University of Washington, 2017.

M. Adams, P. O. Schwartz, H. Johansen, P. Colella, T. J. Ligocki, D. Martin, N. Keen, D. Graves, D. Modiano, B. Van Straalen, et al. Chombo software package for AMR applications-design document. Technical report, 2015.

L. A. Amir, A. Cisternas, W. Dudley, B. G. McAdoo, and G. Pararas-Carayannis. A new tsunami risk scale for warning systems - applications to the Bay of Algiers in Algeria, West Mediterranean Sea. *Journal of Tsunami Society International*, 32(2), 2013.

J. A. Anderson, C. D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10): 5342–5359, 2008.

Applied Technology Council. *Guidelines for Design of Structures for Vertical Evacuation from Tsunamis. Second Edition (FEMA P-646)*. FEMA P-646 Publication, 2012.

M. Arcos and R. J. LeVeque. Validating velocities in the GeoClaw tsunami model using observations near Hawaii from the 2011 Tohoku tsunami. *Pure and Applied Geophysics*, 172(3-4): 849–867, 2015.

H. Arnason. *Interactions between an incident bore and a free-standing coastal structure*. PhD thesis, University of Washington, 2005.

C. Ash. Design of a tsunami vertical evacuation refuge structure in Westport, Washington. In *Structures Congress 2015*, pages 1530–1537, 2015.

B. F. Atwater, S. Musumi-Rokkaku, K. Satake, Y. Tsuji, K. Ueda, and D. K. Yamaguchi. *The orphan tsunami of 1700: Japanese clues to a parent earthquake in North America*. University of Washington Press, 2016.

D. S. Bale, R. J. LeVeque, S. Mitran, and J. A. Rossmanith. A wave propagation method for conservation laws and balance laws with spatially varying flux functions. *SIAM Journal on Scientific Computing*, 24(3):955–978, 2003.

S. Barrachina, M. Castillo, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí. Solving dense linear systems on graphics processors. In *European Conference on Parallel Processing*, pages 739–748. Springer, 2008.

M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

M. J. Berger and I. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1278–1286, 1991.

M. J. Berger, D. L. George, R. J. LeVeque, and K. T. Mandli. The GeoClaw software for depth-averaged flows with adaptive refinement. *Advances in Water Resources*, 34(9):1195–1206, 2011.

M. J. Briggs, C. E. Synolakis, G. S. Harkins, and D. R. Green. Laboratory experiments of tsunami runup on a circular island. *Pure and Applied Geophysics*, 144(3-4):569–593, 1995.

D. Brock. *Understanding Moore's Law: Four Decades of Innovation*. Chemical Heritage Foundation, 2006.

A. R. Brodtkorb, M. L. Sætra, and M. Altinakar. Efficient shallow water simulations on GPUs: Implementation, visualization, verification, and validation. *Computers & Fluids*, 55:1–12, 2012.

G. L. Bryan, M. L. Norman, B. W. O'Shea, T. Abel, J. H. Wise, M. J. Turk, D. R. Reynolds, D. C. Collins, P. Wang, S. W. Skillman, et al. Enzo: An adaptive mesh refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 211(2):19, 2014.

C. Burstedde, L. C. Wilcox, and O. Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

C. Burstedde, G. Stadler, L. Alisic, L. C. Wilcox, E. Tan, M. Gurnis, and O. Ghattas. Large-scale adaptive mantle convection simulation. *Geophysical Journal International*, 192(3):889–906, 2013.

C. Burstedde, D. Calhoun, K. Mandli, and A. R. Terrel. ForestClaw: Hybrid forest-of-octrees AMR for hyperbolic conservation laws. *Parallel Computing: Accelerating Computational Science and Engineering (CSE)*, 25:253–262, 2014.

G. F. Carrier, T. T. Wu, and H. Yeh. Tsunami run-up and draw-down on a plane beach. *Journal of Fluid Mechanics*, 475(March 2002):79–99, 2003.

M. J. Castro, S. Ortega, M. De la Asuncion, J. M. Mantas, and J. M. Gallardo. GPU computing for shallow water flow simulation based on finite volume schemes. *Comptes Rendus Mécanique*, 339(2-3):165–184, 2011.

B. Choi, E. Pelinovsky, K. Kim, and J. Lee. Simulation of the trans-oceanic tsunami propagation due to the 1883 Krakatau volcanic eruption. *Natural Hazards and Earth System Science*, 3(5): 321–332, 2003.

B. H. Choi, D. C. Kim, E. Pelinovsky, and S. B. Woo. Three-dimensional simulation of tsunami run-up around conical island. *Coastal Engineering*, 54(8):618–629, 2007.

R. Cienfuegos, P. A. Catalán, A. Urrutia, R. Benavente, R. Aránguiz, and G. González. What can we do to forecast tsunami hazards in the near field given large epistemic uncertainty in rapid seismic source inversions? *Geophysical Research Letters*, 45(10):4944–4955, 2018.

Clawpack Development Team. Clawpack software. URL http://www.clawpack.org. Version 5.4.0.

C. A. Cornell. Engineering seismic risk analysis. *Bulletin of the seismological society of America*, 58(5):1583–1606, 1968.

D. Cox, T. Tomita, P. Lynett, and R. Holman. Tsunami inundation with macroroughness in the constructed environment. In *Proc. 31st International Conference on Coastal Engineering, ASCE*, pages 1421–1432. World Scientific, 2008.

R. Darlymple and D. Kriebel. Lessons in engineering from the tsunami in Thailand. *The Bridge, Proc.Natl.Acad.Eng*, 35:4–13, 2005.

B. N. Davis and R. J. LeVeque. Adjoint methods for guiding adaptive mesh refinement in tsunami modeling. In *Global Tsunami Science: Past and Future, Volume I*, pages 4055–4074. Springer, 2016.

M. de la Asunción and M. Castro. Simulation of tsunamis generated by landslides using adaptive mesh refinement on GPU. *Journal of Computational Physics*, 345:91–110, 2017.

M. De La Asunción, J. M. Mantas, and M. J. Castro. Simulation of one-layer shallow water systems on multicore and CUDA architectures. *The Journal of Supercomputing*, 58(2):206–214, 2011.

M. de la Asunción, M. J. Castro, E. D. Fernández-Nieto, J. M. Mantas, S. O. Acosta, and J. M. González-Vida. Efficient GPU implementation of a two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes. *Computers & Fluids*, 80: 441–452, 2013.

M. De La Asunción, M. J. Castro, J. M. Mantas, and S. Ortega. Numerical simulation of tsunamis generated by landslides on multiple GPUs. *Advances in Engineering Software*, 99:59–72, 2016.

B. Einfeldt. On Godunov-type methods for gas dynamics. *SIAM Journal on Numerical Analysis*, 25(2):294–318, 1988.

B. Einfeldt, C.-D. Munz, P. L. Roe, and B. Sjögreen. On Godunov-type methods near low densities. *Journal of computational physics*, 92(2):273–295, 1991.

N. C. for Tsunami Research. DART (Deep-ocean Assessment and Reporting of Tsunamis). URL https://nctr.pmel.noaa.gov/Dart/.

B. Fryxell, K. Olson, P. Ricker, F. Timmes, M. Zingale, D. Lamb, P. MacNeice, R. Rosner, J. Truran, and H. Tufo. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273, 2000.

B. Galanti, S. D. Rosen, and A. Salamon. High resolution tsunami modeling at the Mediterranean coast of Israel towards an early warning tsunami scenarios data bank. *Coastal Engineering Proceedings*, 1(32), Feb. 2011.

E. L. Geist and T. Parsons. Probabilistic analysis of tsunami hazards. *Natural Hazards*, 37(3): 277–314, 2006.

D. L. George. *Finite volume methods and adaptive refinement for tsunami propagation and inundation*. Citeseer, 2006.

D. L. George. Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation. *Journal of Computational Physics*, 227(6):3089–3113, 2008.

D. L. George. Adaptive finite volume methods with well-balanced Riemann solvers for modeling floods in rugged terrain: Application to the Malpasset dam-break flood (France, 1959). *International Journal for Numerical Methods in Fluids*, 66(8):1000–1018, 2011.

C. Goldfinger, C. H. Nelson, A. E. Morey, J. E. Johnson, J. R. Patton, E. B. Karabanov, J. Gutierrez-Pastor, A. T. Eriksson, E. Gracia, G. Dunhill, et al. Turbidite event history–methods and implications for Holocene paleoseismicity of the Cascadia subduction zone. Technical report, US Geological Survey, 2012.

F. González, R. LeVeque, and L. Adams. Tsunami hazard assessment of the Ocosta school site in Westport, WA. Technical report, University of Washington, 2013. URL http://hdl.handle.net/1773/24054.

T. R. Hagen, J. M. Hjelmervik, K. A. Lie, J. R. Natvig, and M. O. Henriksen. Visual simulation of shallow-water waves. *Simulation Modelling Practice and Theory*, 13(8):716–726, 2005.

G. P. Hayes and K. P. Furlong. Quantifying potential tsunami hazard in the Puysegur subduction zone, south of New Zealand. *Geophysical Journal International*, 183(3):1512–1524, 2010.

R. D. Hornung and S. R. Kohn. Managing application complexity in the SAMRAI object-oriented framework. *Concurrency and computation: practice and experience*, 14(5):347–368, 2002.

J. Horrillo, S. T. Grilli, D. Nicolsky, V. Roeber, and J. Zhang. Performance benchmarking tsunami models for NTHMP's inundation mapping activities. *Pure and Applied Geophysics*, 172(3-4): 869–884, 2014.

K. Hu, C. G. Mingham, and D. M. Causon. Numerical simulation of wave overtopping of coastal structures using the non-linear shallow water equations. *Coastal engineering*, 41(4):433–465, 2000.

M. E. Hubbard and N. Dodd. A 2D numerical model of wave run-up and overtopping. *Coastal Engineering*, 47(1):1–26, 2002.

F. Imamura, S. Koshimura, and K. Goto. Global disaster : The 2004 Indian Ocean tsunami. *Journal of Disaster Research*, 1(1):131–135, 2007.

W. Johnstone and B. Lence. Assessing the value of mitigation strategies in reducing the impacts of rapid-onset, catastrophic floods. *Journal of Flood Risk Management*, 2(3):209–221, 2009.

U. Kânoğlu and C. E. Synolakis. Long wave runup on piecewise linear topographies. *Journal of Fluid Mechanics*, 374:1–28, 1998.

A. Lacasta, M. Morales-Hernández, J. Murillo, and P. García-Navarro. GPU implementation of the 2D shallow water equations for the simulation of rainfall/runoff events. *Environmental Earth Sciences*, 74(11):7295–7305, 2015.

M. Lastra, J. M. Mantas, C. Ureña, M. J. Castro, and J. A. García-Rodríguez. Simulation of shallow-water systems using graphics processing units. *Mathematics and Computers in Simulation*, 80(3):598–618, 2009.

W. Leng and S. Zhong. Implementation and application of adaptive mesh refinement for thermo-chemical mantle convection studies. *Geochemistry, Geophysics, Geosystems*, 12(4), 2011.

L. J. Leonard, C. A. Currie, S. Mazzotti, and R. D. Hyndman. Rupture area and displacement of past Cascadia great earthquakes from coastal coseismic subsidence. *Bulletin*, 122(11-12): 2079–2096, 2010.

R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics*, 131(2):327–353, 1997.

R. J. LeVeque, D. L. George, and M. J. Berger. Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, 20:211–289, 2011.

R. J. LeVeque, P. Bodin, G. Cram, et al. Developing a warning system for inbound tsunamis from the Cascadia Subduction Zone. 2018. URL http://staff.washington.edu/rjl/pubs/Oceans2018.

P. Lin and P. L. F. Liu. A numerical study of breaking waves in the surf zone. *Journal of Fluid Mechanics*, 359:239–264, 1998.

J. W. V. D. Lindt, R. Gupta, D. T. Cox, and J. S. Wilson. Wave impact study on a residential building. *Journal of Disaster Research*, 4:419–426, 2009.

P. L.-F. Liu, P. Lynett, H. Fernando, B. E. Jaffe, H. Fritz, B. Higman, R. Morton, J. Goff, and C. Synolakis. Observations by the international tsunami survey team in Sri Lanka. *Science (New York, N.Y.)*, 308(5728):1595, jun 2005.

W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig. Molecular dynamics simulations on commodity GPUs with CUDA. In *International Conference on High-Performance Computing*, pages 185–196. Springer, 2007.

P. J. Lynett. Effect of a shallow water obstruction on long wave runup and overland flow velocity. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 133(6):455–462, 2007.

P. J. Lynett, J. A. Melby, and D. H. Kim. An application of Boussinesq modeling to hurricane wave overtopping and inundation. *Ocean Engineering*, 37(1):135–153, 2010.

J. Macías, A. Mercado, J. M. González-Vida, S. Ortega, and M. J. Castro. Comparison and computational performance of Tsunami-HySEA and MOST models for LANTEX 2013 scenario: Impact assessment on Puerto Rico coasts. In *Global Tsunami Science: Past and Future, Volume I*, pages 3973–3997. Springer, 2016.

B. T. MacInnes, A. R. Gusman, R. J. LeVeque, and Y. Tanioka. Comparison of earthquake source models for the 2011 Tohoku event using tsunami simulations and near field observations. *Bull. Seis. Soc. Amer.*, pages 1256–1274, 2013.

P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, and C. Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer physics communications*, 126(3):330–354, 2000.

K. T. Mandli and C. N. Dawson. Adaptive mesh refinement for storm surge. *Ocean Modelling*, 75: 36–50, Mar. 2014.

K. T. Mandli, A. J. Ahmadia, M. Berger, D. Calhoun, D. George, Y. Hadjimichael, D. I. Ketcheson, G. I. Lemoine, and R. J. LeVeque. Clawpack: building an open source ecosystem for solving hyperbolic PDEs. *PeerJ Computer Science*, 2:e68, 2016.

S. Mayer and P. A. Madsen. Simulation of breaking waves in the surf zone using a Navier-Stokes solver. In *Coastal Engineering 2000*, pages 928–941. 2001.

D. Melgar, R. J. LeVeque, D. S. Dreger, and R. M. Allen. Kinematic rupture scenarios and synthetic displacement data: An example application to the Cascadia Subduction Zone. *Journal of Geophysical Research: Solid Earth*, 121(9):6658–6674, 2016.

F. Menter. Zonal two equation kw turbulence models for aerodynamic flows. In *23rd fluid dynamics, plasmadynamics, and lasers conference*, page 2906, 1993.

F. Menter and T. Esch. Elements of industrial heat transfer predictions. In *16th Brazilian Congress of Mechanical Engineering (COBEM)*, volume 109, page 650. sn, 2001.

F. R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1):625–632, 2003.

J. Michalakes and M. Vachharajani. GPU acceleration of numerical weather prediction. *Parallel Processing Letters*, 18(04):531–548, 2008.

O. e. Moctar, V. Shigunov, and T. Zorn. Duisburg test case: Post-panamax container ship for benchmarking. *Ship Technology Research*, 59(3):50–64, 2012.

N. Mori, T. Takahashi, T. Yasuda, and H. Yanagisawa. Survey of 2011 Tohoku earthquake tsunami inundation and run-up. *Geophysical Research Letters*, 38(18):6–11, 2011.

M. Motley, G. Lemoine, and S. Livermore. Three-dimensional loading effects of tsunamis on bridge superstructures. In *Structures Congress 2014*, pages 1348–1358, 2014.

M. R. Motley, H. K. Wong, X. Qin, A. O. Winter, and M. O. Eberhard. Tsunami-induced forces on skewed bridges. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 142(3):04015025, 2015.

A. Muhari, F. Imamura, S. Koshimura, and J. Post. Examination of three practical run-up models for assessing tsunami impact on highly populated areas. *Natural Hazards and Earth System Science*, 11(12):3107–3123, 2011.

J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming with CUDA. In *ACM SIGGRAPH 2008 classes*, page 16. ACM, 2008.

M. Nosov. Tsunami waves of seismic origin: The modern state of knowledge. *Izvestiya, Atmospheric and Oceanic Physics*, 50(5):474–484, 2014.

NVIDIA. CUDA C Programming Guide. URL https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

Y. Okada. Surface deformation due to shear and tensile faults in a half-space. *Bulletin of the seismological society of America*, 75(4):1135–1154, 1985.

C. Ozer Sozdinler, A. C. Yalciner, and A. Zaytsev. Investigation of tsunami hydrodynamic parameters in inundation zones with different structural layouts. *Pure and Applied Geophysics*, 172 (3-4):931–952, 2015.

H. Park, D. T. Cox, P. J. Lynett, D. M. Wiebe, and S. Shin. Tsunami inundation modeling in constructed environments: A physical and numerical comparison of free-surface elevation, velocity, and momentum flux. *Coastal Engineering*, 79:9–21, 2013.

M. D. Petersen, C. H. Cramer, and A. D. Frankel. Simulations of seismic hazard for the Pacific Northwest of the United States from earthquakes associated with the Cascadia Subduction Zone. In *Earthquake Processes: Physical Modelling, Numerical Simulation and Data Analysis Part I*, pages 2147–2168. Springer, 2002.

A. Plus. Number of injured in Indonesia tsunami surges to over 14,000, Dec 2018. URL https://www.thestar.com.my/news/regional/2018/12/31/number-of-injured-in-indonesia-tsunami-surges-to-over-14000/.

S. B. Pope. Turbulent flows, 2001.

N. Pophet, N. Kaewbanjak, J. Asavanant, and M. Ioualalen. High grid resolution and parallelized tsunami simulation with fully nonlinear Boussinesq equations. *Computers & Fluids*, 40(1): 258–268, 2011.

S. Popinet. Adaptive modelling of long-distance wave propagation and fine-scale flooding during the Tohoku tsunami. *Natural Hazards and Earth System Science*, 12(4):1213–1227, 2012.

X. Qin, M. R. Motley, R. J. LeVeque, and F. I. Gonzalez. Multi-scale modeling of a 500-year CSZ tsunami inundation with constructed environment. In *Computing in Civil Engineering 2017*, pages 318–325. 2017.

X. Qin, R. J. LeVeque, and M. R. Motley. Accelerating wave-propagation algorithms with adaptive mesh refinement using the Graphics Processing Unit (GPU). *arXiv preprint arXiv:1808.02638*, 2018a.

X. Qin, M. Motley, R. LeVeque, F. Gonzalez, and K. Mueller. A comparison of a two-dimensional depth-averaged flow model and a three-dimensional RANS model for predicting tsunami inundation and fluid forces. *Natural Hazards & Earth System Sciences*, 18(9), 2018b.

X. Qin, M. R. Motley, and N. A. Marafi. Three-dimensional modeling of tsunami forces on coastal communities. *Coastal Engineering*, 140:43–59, 2018c.

X. Qin, R. J. LeVeque, and M. R. Motley. Accelerating an adaptive mesh refinement code for depth-averaged flows using Graphics Processing Units (GPUs). *Journal of Advances in Modeling Earth Systems*, 2019.

J. D. Ramsden. *Tsunamis: forces on a vertical wall caused by long waves, bores, and surges on a dry bed*. PhD thesis, California Institute of Technology, 1993.

Z.-y. Ren, B.-l. Wang, T.-t. Fan, and H. Liu. Numerical analysis of impacts of 2011 Japan Tohoku tsunami on China Coast. *Journal of Hydrodynamics, Ser. B*, 25(4):580–590, Sept. 2013.

P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

M. Rueben, R. Holman, D. Cox, S. Shin, J. Killian, and J. Stanley. Optical measurements of tsunami inundation through an urban waterfront modeled in a large-scale laboratory basin. *Coastal Engineering*, 58(3):229–238, 2011.

M. C. Schatz, C. Trapnell, A. L. Delcher, and A. Varshney. High-throughput sequence alignment using Graphics Processing Units. *BMC bioinformatics*, 8(1):474, 2007.

H.-Y. Schive, Y.-C. Tsai, and T. Chiueh. GAMER: a graphic processing unit accelerated adaptive-mesh-refinement code for astrophysics. *The Astrophysical Journal Supplement Series*, 186(2): 457, 2010.

H.-y. Schive, U.-h. Zhang, and T. Chiueh. Directionally unsplit hydrodynamic schemes with hybrid MPI/OpenMP/GPU parallelization in AMR. *The International Journal of High Performance Computing Applications*, 26(4):16, 2011.

F. Shi, J. T. Kirby, J. C. Harris, J. D. Geiman, and S. T. Grilli. A high-order adaptive time-stepping TVD solver for Boussinesq modeling of breaking waves and coastal inundation. *Ocean Modelling*, 43-44:36–51, 2012.

S. Shin, K.-H. Lee, H. Park, D. T. Cox, and K. Kim. Influence of a infrasturcture on tsunami inundation in a coastal city. *Coastal Engineering*, pages 1–10, 2012.

L. S. Smith and Q. Liang. Towards a generalised GPU/CPU shallow-flow modelling tool. *Computers and Fluids*, 88:334–343, 2013.

D. Spalding. A single formula for the law of the wall. *Journal of Applied Mechanics*, 28(3): 455–458, 1961.

I. A. Svendsen. Analysis of surf zone turbulence. *Journal of Geophysical Research: Oceans*, 92 (C5):5115–5124, 1987.

C. E. Synolakis. The runup of solitary waves. *Journal of Fluid Mechanics*, 185(-1):523, 1987.

The OpenFOAM Foundation. OpenFOAM v2.1.0: Multiphase Modelling. http://www.openfoam.org/version2.1.0/, 2014a.

The OpenFOAM Foundation. OpenFOAM v2.3.1. http://www.openfoam.org/version2.3.1/, 2014b.

The OpenFOAM Foundation. Predictor-Corrector Semi-Implicit MULES. http://openfoam.org/release/2-3-0/multiphase/, 2014c.

V. Titov, D. Arcas, C. Moore, R. LeVeque, L. Adams, and F. González. Tsunami hazard assessment of Bainbridge Island, Washington project report. Submitted to Washington State Emergency Management Division and Department of Natural Resources, 2018. URL http://staff.washington.edu/rjl/pubs/THA_Bainbridge.

V. V. Titov and F. I. Gonzalez. Implementation and testing of the method of splitting tsunami (MOST) model, 1997.

V. V. Titov and C. E. Synolakis. Numerical modelling of tidal wave runup. *ASCE Journal of Waterway, Port, Coastal and Ocean Engineering*, 124(4):157–171, 1998.

V. V. Titov, F. I. Gonzalez, E. Bernard, M. C. Eble, H. O. Mofjeld, J. C. Newman, and A. J. Venturato. Real-time tsunami forecasting: Challenges and solutions. *Natural Hazards*, 35(1): 35–41, 2005.

T. Tomita and K. Honda. Tsunami estimation including effect of coastal structures and buildings by 3D model. *Coastal Structures*, 2007.

T. Tomita, K. Honda, and T. Kakinuma. Application of storm surge and tsunami simulator in oceans and coastal areas (stoc) to tsunami analysis. In *Joint Panel Conference of the US-Japan Cooperative Program in Natural Resources*, volume 38, pages 109–115, 2006.

Tsunami Pilot Study Working Group and others. Seaside, Oregon tsunami pilot study-modernization of FEMA flood hazard maps. Technical report, U.S. Geological Survey, 2006.

M. D. Turzewski, K. W. Huntington, and R. J. LeVeque. The geomorphic impact of outburst floods: Integrating observations and numerical simulations of the 2000 Yigong flood, eastern Himalaya. *Journal of Geophysical Research: Earth Surface*, 2019.

University of Southern California. NTHMP mapping and modeling benchmarking wrokshop: Tsunami currents, 2015.

USGS. Seismic design maps and tools. http://earthquake.usgs.gov/hazards/designmaps/ [Accessed: 2016-03-07], 2016.

P. Wang, T. Abel, and R. Kaehler. Adaptive mesh fluid simulations on GPU. *New Astronomy*, 15 (7):581–589, 2010.

Y. Wei, E. N. Bernard, L. Tang, R. Weiss, V. V. Titov, C. Moore, M. Spillane, M. Hopkins, and U. Kânoğlu. Real-time experimental forecast of the Peruvian tsunami of August 2007 for US coastlines. *Geophysical Research Letters*, 35(4), 2008.

Y. Wei, C. Chamberlin, V. V. Titov, L. Tang, and E. N. Bernard. Modeling of the 2011 Japan

tsunami: Lessons for near-field forecast. *Pure and Applied Geophysics*, 170(6-8):1309–1331, 2013.

M. Wei-Haas. The Science of Indonesia's Surprise Tsunami, Oct 2018. URL https://www.nationalgeographic.com/environment/2018/09/indonesia-tsunami-sulawesi-explained-science-geology/?cmpid=org=ngp::mc=social::src=twitter::cmp=editorial::add=tw20180928env-indonesiatsunamiupdate::rid=&sf198797445=1.

I. A. Williams and D. R. Fuhrman. Numerical simulation of tsunami-scale wave boundary layers. *Coastal Engineering*, 110:17–31, 2016.

A. O. Winter, M. R. Motley, and M. O. Eberhard. Tsunami-like wave loading of individual bridge components. *Journal of Bridge Engineering*, 23(2):04017137, 2017.

H. Xiao and W. Huang. Numerical modeling of wave runup and forces on an idealized beachfront house. *Ocean Engineering*, 35(1):106–116, 2008.

V. Yakhot, S. Orszag, S. Thangam, T. Gatski, and C. Speziale. Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics (1989-1993)*, 4(7):1510–1520, 1992.

H. Yeh. Tsunami forces in the runup zone. *Caribbean Tsunami Hazard - Proceedings of the NSF Caribbean Tsunami Workshop*, 132(December):275–287, 2006.

H. Yeh. Design tsunami forces for onshore structures. *Journal of Disaster Research*, 2(6):531–536, 2007.

H. Yeh, S. Sato, and Y. Tajima. The 11 March 2011 East Japan earthquake and tsunami: Tsunami effects on coastal infrastructure and buildings. *Pure and Applied Geophysics*, 170(6-8):1019–1031, 2013.

W. Zhang, A. Almgren, M. Day, T. Nguyen, J. Shalf, and D. Unat. Boxlib with tiling: An adaptive mesh refinement software framework. *SIAM Journal on Scientific Computing*, 38(5):S156–S172, 2016.

Y. J. Zhang and A. M. Baptista. An efficient and robust tsunami model on unstructured grids. Part I: Inundation benchmarks. *Pure and Applied Geophysics*, 165(11-12):2229–2248, 2008.

Appendix A

COMPUTATION OF SEISMIC LOAD

The design earthquake loads for Seaside, Oregon are computed using the ASCE 7-10 Standard [American Society of Civil Engineers (ASCE), 2013]. Building I was assumed to be founded on very dense soil and soft rock (Site Class C according to ASCE 7-10 Table 20.3-1). The building's occupancy category is classified as low risk to human life (Risk Category I according to ASCE 7-10 Table 1.5-1). Using the Risk-Adjusted Maximum Considered Earthquake maps (ASCE 7-10 Chapter 22 and the USGS web tool [USGS, 2016]), the mapped short-period spectral acceleration, S_s is 1.33g and the mapped 1-second spectral accelerations, S_1 is 0.68g.

Those spectral accelerations are adjusted for local site conditions using ASCE 7-10 Eq. 11.4-1 (A.1) and Eq. 11.4-2 (A.2). Where the short-period site adjustment factor F_a is found to be 1.0 (ASCE 7-10 Table 11.4-1) and the 1-second period site adjustment factor F_v is equal to 1.3 (ASCE 7-10 Table 11.4-2).

$$S_{MS} = F_a S_S \tag{A.1}$$

$$S_{M1} = F_v S_1 \tag{A.2}$$

The maximum considered spectral accelerations (S_{MS} and S_{M1}) are then adjusted for design. The short-period design spectral acceleration parameter, S_{DS} is computed as 0.89g using ASCE 7-10 Eq. 11.4-3 (A.3) and the long-period design spectral acceleration parameter, S_{D1} is computed as 0.59g using ASCE 7-10 Eq. 11.4-4 (A.4).

$$S_{DS} = \frac{2}{3} S_{MS} \tag{A.3}$$

$$S_{D1} = \frac{2}{3} S_{M1} \tag{A.4}$$

The building's lateral force resisting system is assumed to be special reinforced concrete shear walls with a response modification coefficient, R, equal to 5 (according to ASCE 7-10 Table 12.2-1). The structure's height is assumed to be 13.7m from the ground level. The code approximated fundamental period of the structure, T_a is estimated as 0.35s using ASCE 7-10 Equation 12.8-7 (A.5), where C_t and x are found in ASCE 7-10 (Table 12.8-2) and are equal to 0.0488 and 0.75 respectively.

$$T_a = C_t h_n^x \tag{A.5}$$

The fundamental period of the structure for design, T, shall not exceed the upper limit on the calculated period, $C_u T_a$, computed as 0.49s (where C_u is found in ASCE 7-10 Table 12.8-1). In this study, we assume that the actual fundamental period of the structure is between T_a and the upper limit on the calculated period.

The seismic design base shear, V, is computed as per the Equivalent Lateral Force Procedure (ASCE 7-10 Section 12.8) using ASCE 7-10 Eq. 12.8-1 (A.6). Where C_s is the seismic response coefficient, and W is the effective seismic weight of the building.

$$V = C_s W \tag{A.6}$$

The value of C_s is computed using ASCE 7-10 Eq. 12.8-2 (A.7) and for the case where $T \leq T_L$, C_s shall not exceed the value computed using ASCE 7-10 Eq. 12.8-3 (A.7). The long-period transition period, T_L, is equal to 16s (ASCE 7-10 Figure 22-12). The value of C_s shall also be more than the values computed using ASCE Eq. 12.8-5 (A.9) and Eq. 12.8-6 (A.10). In Eq. A.7 through Eq. A.10, I_e is equal to 1.0 for Risk Category I (according to ASCE 7-10 Table Table 1.5-2). The final value of C_s is computed as 0.18g.

$$C_s = \frac{S_{DS}}{\frac{R}{I_e}} \tag{A.7}$$

$$C_s = \frac{S_{D1}}{T\frac{R}{I_e}} \tag{A.8}$$

$$C_s = 0.044 S_{DS} I_e \geq 0.01 \tag{A.9}$$

$$C_s = 0.05 S_1 / (R/I_e) \tag{A.10}$$

The average seismic weight of the each floor is assumed to be around $9.6 kN/m^2$. The building is 4 stories tall with a floor plate that is approximately 12.2 by 39.0 m. The estimated total building seismic weight, W, is around 18,200 kN. The seismic design base shear, V, is estimated to 3,350 kN.