The Automation of Numerical Models of Coseismic Tsunamis

Codi Allen Wiersma

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Geosciences

Robert Weiss, Chair
Martin Chapman
D S. Stamps
Timothy Warburton

July 12, 2019
Blacksburg, Virginia

Keywords: Numerical Modeling, Tsunami, GeoClaw, Finite Volume

The Automation of Numerical Models of Coseismic Tsunamis

Codi Allen Wiersma

ABSTRACT

The use of tsunami models for applications of 'now-casting', which is the prediction of the present and near future behavior, has limited exploration, and could potentially be of significant usefulness. Tsunamis are most often caused by earthquakes in subduction zones, which generates coupled uplift and subsidence, and displaces the water column. The behavior of the fault failure is difficult to describe in the short term, often requiring seismic waveform inversion, which takes a length of time on the order of weeks to months to properly model, and is much too late for any use in a now-casting sense. To expedite this length of time, a series of source models are created with variable fault geometry behaviors, using fault parameters from Northern Oceanic and Atmospheric Administration's Short-term Inundation and Forecasting of Tsunamis (SIFT) database, in order to model a series of potential tsunami behaviors using the numerical modelling package, GeoClaw. The implementation of modeling could identify areas of interest for further study that are sensitive to fault failure geometry. Initial results show that by varying the geometry of sub-faults of a given earthquake, the resulting tsunami models behave fairly differently with different wave dispersion behavior, both in pattern and magnitude. While there are shortcomings of the potential geometries the code created in this study, and there are significant improvements that can be made, this study provides a good starting point into now-casting of tsunami models, with future iterations likely involving statistical probability in the fault failure geometries.

The Automation of Numerical Models of Coseismic Tsunamis

Codi Allen Wiersma

GENERAL AUDIENCE ABSTRACT

Short term modeling of tsunamis generated by earthquakes is poorly explored. If an earthquakes causes movement in a fault located underwater, and this movement will then cause the water column above it to be displaced. Tsunami models are sensitive to how the fault moves, and an accurate representation of this movement often takes much more time that the duration of a tsunami. This lengthy process is ineffective for shot term modeling. This study instead estimates several possible scenarios of how the fault will behave, and model each of them. This will show how different locations of interest are sensitive to different geometries of fault failure. Initial results show that by varying this geometry, the tsunami wave behaves very differently, and will cause different amounts of run-up in the same location depending on which particular geometry is modeled. The automation of distinctly different earthquake sources serve as a good starting point for future work to be conducted to generate more accurate models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Approximately 10 percent of the worlds population lives lower than 10 m above sea level, with 40 percent of the population living within 100 km of the coast[1]. The ocean economy is also estimated by the UN to be between 3 and 6 trillion dollars per year USD. Both the global economy as well as the health of individuals are highly affected by events that occur on our coastlines. One of the most damaging events that can occur at our coastlines are tsunamis. The total cost in property damage due to hurricanes during the entire year of 2017, the most significant cause of natural disaster costs in the United States, is estimated to be $306.3 billion, breaking the previous record of $214.8 billion from 2005[2]. To compare, the Tohoku tsunami in 2011 caused $227 billion dollars in damages (Kazama & Noda, 2012). A single tsunami can be as costly as a years worth of hurricanes.

Tsunamis are a sequence of long wavelength water waves in deep ocean, and shorten as they approach shallow water. This increases the amplitude of the wave, resulting in the large waves. Tsunamis are caused by the displacement of a significant amount of water.

[1] https://www.un.org/sustainabledevelopment/wpcontent/uploads/2017/05/Ocean-fact-sheet-package.pdf

[2] https://coast.noaa.gov/states/fast-facts/hurricane-costs.html

This displacement can be caused by a variety of naturally occurring events:

- Asteroid impacts can generate significant tsunami waves. This is evidenced by Hills (1997), which cites two locations, one in Hawaii and a second in Alaska, with wave height evidence of greater than 300 meters. The Cretacious-Tertiary impact caused tsunami wave heights measuring between 50 and 100 meters in locations in Texas. Despite the significant amounts of damage that these events would cause, they are extremely rare, with 12 in the last 5 million years capable of generating any size tsunami, and are unlikely to occur in the near future (Hills & Mader, 1997).

- Volcanic eruptions and caldera collapses are capable of generating tsunamis, although to a lesser extent than many other sources. The famed Krakatoa eruption of 1883 was only capable of generating a 2 meter tsunami wave, despite being one of the most violent eruptions in recorded history (Latter, 1981). These events are also rare, as either a significant eruption or a flank collapse is required to displace enough water to create a long wavelength water wave.

- Landslides and submarine mass failures (underwater landslides) are the second most common source of tsunamis, and are often used in the validation of tsunami models, as seen in Gonzalez et al., 2011. A large volume of earth displacing water can generate a tsunami. An example of this can be seen in Alaska in 1994, when a submarine mass failure caused a tsunami with a run-up of 11 meters (Kulikov, Rabinovich, Thomson, & Bornhold, 1996).

- Earthquakes are the most common cause of tsunamis. They occur more frequently than the other sources of tsunamis. An earthquake causes deformation in the rock that, if of sufficient size and in an ocean basin, can displace a large volume of water above the deformation, in turn generating a tsunami. Large earthquakes are rela-

> tively common, with one earthquake greater than an 8.0 magnitude in eight of the last ten years, and with earthquake magnitudes between 7.0 and 8.0 occurring an average of 15 times during the same years.

Tsunamis are capable of travelling at 800 km/hr. This speed is dependent on wave height and water depth, however. An increase in wave height or a decrease in water depth both decrease the velocity of a tsunami down to about 45 km/hr. According to the USGS, [3] an earthquake with a magnitude greater than 7.8 can displace enough water to generate a tsunami. Below 7.8 M, but above 7.5 M can create tsunamis that are dangerous to localities within one or two hundred kilometers, but less so to further locations. Earthquakes between 6.5 M and 7.5 M often are not strong enough to generate a tsunami, but are capable of generating landslides. An example of this occurred in 1958 in Lituya Bay, Alaska, where a 7.5 magnitude earthquake created a landslide within the bay. This landslide caused wave run-up in excess of 520 meters on the opposite side of the bay, and 30 meters throughout the bay (Mader & Gittings, 2002).

1.1 Purpose and Outline

The purpose of this study is to automate the numerical modelling of tsunamis generated by earthquakes to quickly estimate the initial fault parameters of an earthquake for implementation in a 'nowcasting' type method, where the earthquake which is assumed to generate a tsunami has occurred minutes to hours beforehand. The more common approach to numerical modelling of tsunamis is to design initial fault parameters of an earthquake. This can be fairly time consuming, and is dependent upon good source models of

[3] `https://www.usgs.gov/faqs/what-it-about-earthquake-causes-a-tsunami?qt-news_science_products=0#qt-news_science_products`

a given earthquake. This estimation of earthquake source parameters often relies on seismic waveform inversion, as described by Johnson (1998), and requires a significant amount of time and slows the modelling process by days or longer. By estimating the parameters of a given earthquake, a faster, but a less accurate model can be generated. Estimating several source parameters and conducting different models can be done easily, allowing for initial source estimations of a given earthquake using only the location of the epicenter and the magnitude of said earthquake as the input from the user. This will allow for the determination of a study area's sensitivity to an earthquake's source parameters as well as the resulting sea floor and water column deformation, and after empirical results are compared to the model results, it is possible to determine the most accurate initial earthquake parameters.

This thesis is organized as follows:

- Chapter 2 is a literature review with the intent to give the reader some foresight into the code used to model tsunami propagation. The three main themes explored are the validity of GeoClaw, which is explored in two different studies, the sensitivity of co-seismic tsunami propagation to earthquake fault parameters and then explore the use of 'sub-faults' as detailed by the National Oceanic and Atmospheric Administration's (NOAA) Center for Tsunami Research [4].

- Chapter 3 describes the implementation of the numerical modelling package GeoClaw which models tsunami propagation through a finite volume method. GeoClaw relies on the shallow water equations, and is improved greatly through the implementation of adaptive mesh refinement, both of which are elaborated upon in the chapter. The modifications made to the program, as well as the new Python scripts created, are explained in this chapter in detail, such as the automation of fault ge-

[4]https://nctr.pmel.noaa.gov/index.html

ometry constraining, the implementation of downloading and meshing bathymetry grids across the anti-meridian.

- Chapter 4 follows two case studies of the implementation of the modifications made to the GeoClaw code and the affects that variations in source geometry will have on wave height in a desired location. The first study area is of the Chilean 2010 earthquake, which had a magnitude of 8.8 and an epicenter location of 35.909 S and 72.733 W. This location was chosen due to the example given in GeoClaw when it is first installed. The second study area is of the 2011 Tohoku-Oki Earthquake, which had a magnitude between 9.0 and 9.1 with an epicenter location of 38.322 N and 142.369 E. This study area was chosen as it is one of the most devastating earthquakes and resulting tsunami in recent history.

- Chapter 5 addresses the conclusions of this thesis, and the validity of using this process for initial earthquake fault parameter estimation.

- Chapter 6 will address the expected future works in relation to the implementation of this research. This will include the implementation of a probability distribution based around the initial fault geometries, applying a distribution of slip rather than assuming a uniform slip on all sub-faults, and variations in magnitude and epicenter locations, as initial variables can be difficult to predict soon after an event.

Chapter 2

Literature Overview

The Conservation Law Package (Clawpack) is a free, open source software package that has been under development since 1994 (Berger, George, LeVeque, & Mandli, 2011). Clawpack is used to solve hyperbolic systems of partial differential equations in up to three special dimensions, and it includes an adaptive mesh refinement that allows for the easy implementation of hyperbolic problems. The adaptive mesh refinement is explained in more detail in Berger & LeVeque (1998), which is explored in Section 3.1.2. GeoClaw, a subset of Clawpack that corrects for limitations of Clawpack, is capable of modelling geophysical flows (Berger et al., 2011). These limitations include flows which take place over topography or bathymetry, and can be specified in either a single or in multiple overlapping data sets with either the same or different resolutions. It also has an option to use longitude-latitude grids rather than using Cartesian grid systems. GeoClaw also allows for the existence of a nontrivial steady state, an ocean at rest for example, as often the intention of the model is to show a perturbation in the steady state; e.g. a tsunami interacting with the ocean (LeVeque, George, & Berger, 2011).

GeoClaw is a finite-volume wave propagation numerical modeling code, explained more

thoroughly in Chapter 3, that models various fluid flow problems through the solving of the non-linear free surface shallow water equations, which are shown by Equation 3.1, describing the Shallow Water Equations. These equations are generally considered appropriate for tsunami modelling because they can accurately describe wavelengths required to create tsunamis, which are long in comparison to the depth of the ocean (Berger et al., 2011). GeoClaw uses Riemann solvers to handle wave propagation and inundation through the use of adaptive mesh refinement (LeVeque et al., 2011). The Riemann solvers are also expanded upon further in Section 3.1.2. GeoClaw is capable of modeling a series of flow problems, such as storm surges (Mandli & Dawson, 2014), dam breaks (D. George, 2011), as shown in Figure 2.3, debris flows (D. L. George & Iverson, 2011), and tsunami propagation (LeVeque et al., 2011), with the majority of the program written in Fortran, and a user interface written in Python (Berger et al., 2011).

Figure 2.1: A map of the south east of France, showing the relative location of the town of Frejus, and the Reyran River Valley to the north

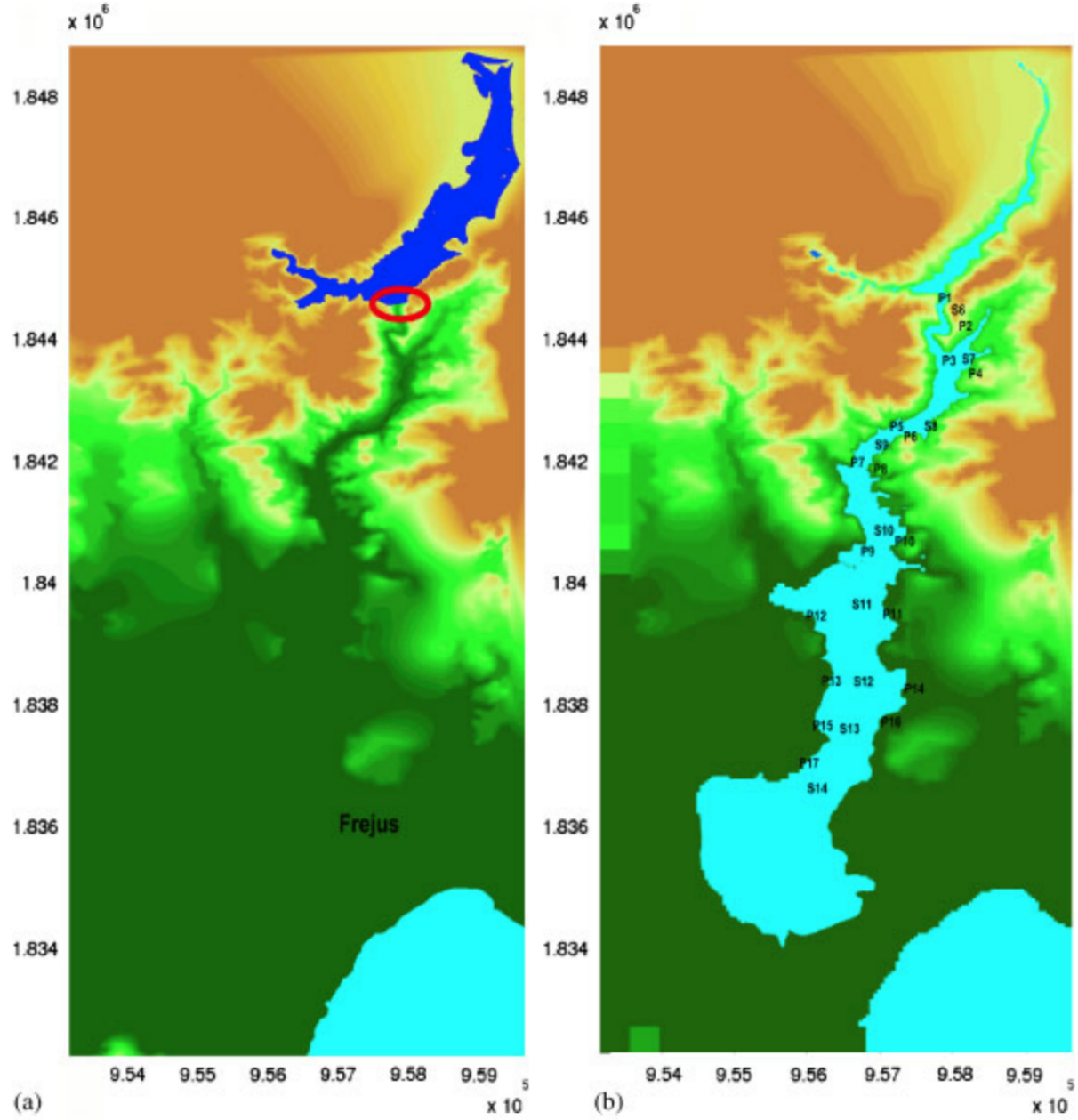


Figure 2.2: The study area used in D. George 2011. The Malpassat Dam can be seen to the north east, circled in red, and the town of Frejus labelled, at 43.433 N, 6.737 E, used in the George (2011). The figure has a range of 6.47 km in the East-West direction, and 16.58 km in the North-South. Adapted from D. George 2011.

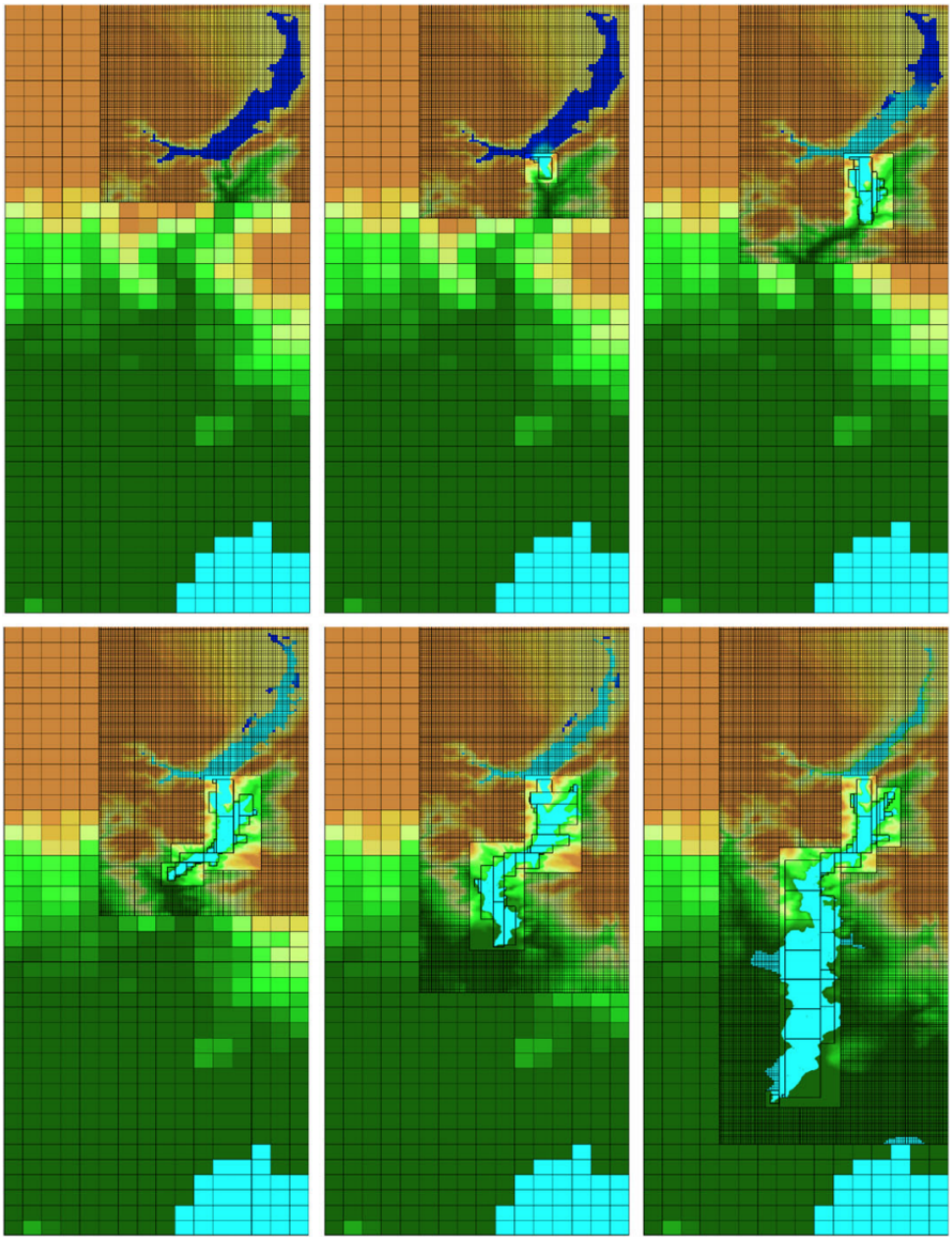


Figure 2.3: The Malpasset Dam failure in France 1959, is the same study area from 2.2, now modeled in GeoClaw. The top left figure shows the topography of the Reyran River Valley before the flood. Each following figure shows the progression of a GeoClaw model, where in the bottom right the reservoir has drained considerably, and the nearby town of Frejus has been inundated.

2.1 Validation of GeoClaw

An important question to consider is whether these codes produce valid results when compared to empirical evidence. Gonzalez et al., 2011 tests the validity of the GeoClaw model through a series of tsunami benchmark validation tests from the National Tsunami Hazards Mitigation Program (NTHMP). Of the 9 tests conducted, 8 were artificial laboratory tests conducted within numerical models, with the final test comparing a model generated by GeoClaw to gathered field data on the 1993 Okushiri Island tsunami. Of the 8 artificial tests, GeoClaw passed 6, with one of the remaining two tests violating assumptions of the shallow water equation solutions. The first failure is listed as benchmark number 4, and is caused by the use of the shallow water wave equation, which assumes that the wave height is significantly small compared to the depth of the water. In this test, a single wave approaching a beach is simulated in a wave tank, shown in Figure 2.5. In cases where the height of the wave did not exceed about 30% of the depth of the tank, the tests seem to correlate well. Once this limit is exceeded, however, the test fails (González et al., 2011). The second benchmark, listed in the study as number 8, likely does not create a long wavelength wave, as it used the movement of a single large block to represent a submarine mass failure. To do this, a change in bathymetry was coded to mimic a mass failure that did not disperse. GeoClaw was able to model this phenomenon, but the tide gauges of the test were noisy and no longer correlated well. This is most likely due to the unrealistic piece-wise style change of bathymetry and a high probability that the mass failure did not produce wavelengths sufficiently long enough to make the assumptions of the shallow water equations valid. The results of the model compared to the field data from the Okushiri tsunami were mixed. Overall the general behavior of the model conformed well with observations but a lack of highly accurate bathymetry and topography data combined with vague information in the field data limits the comparison between the two.

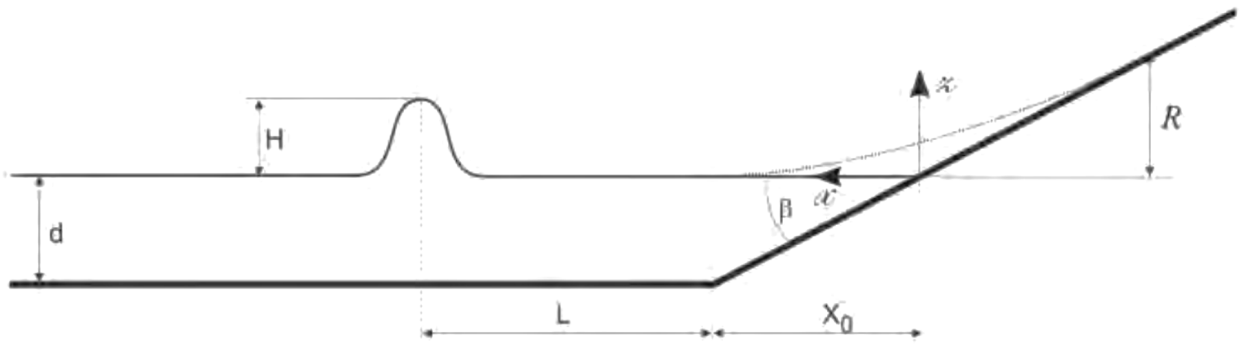


Figure 2.4: The computational failure of GeoClaw in Benchmark 6 in Gonzalez et al., 2011, is shown above. d represents the depth of the water, H is the height of the approaching wave, R is the maximum run-up distance, L is the distance from the wave to the start of the incline, X_0 is the distance from the start of the incline to the intersection of the water surface, and beta is the angle of the incline of the beach. The test fails to accurately represent GeoClaw due to the exceeding of design assumptions of the Shallow Wave Equations in that H exceeds a significant ratio of d (eg H/d). The work of Gonzalez et al., 2011 shows that this limit appears to be approximately 30 percent, while the benchmark example produced a wave height that was 43 percent of the water depth. If this ratio is exceeded, then the shallow water equation assumptions in GeoClaw begin to break down and no longer properly model tsunami propagation.

Another study more recently compares the velocities calculated in GeoClaw to observations in Hawaii caused by the 2011 Tohoku tsunami (Arcos & LeVeque, 2015), as seen by Figure 2.6. At the time of the tsunami, there were 328 current meters around the Hawaiian Islands in 18 different locations, as shown on Figure 2.6, that captured the time series of fluid velocity and wave height. These stations captured a range of bathymetric conditions, from coastlines to deep channels and harbors. A GeoClaw model was conducted using a 1 min resolution using ETOPO1 bathymetry, and modeled currents were recorded at 10 of the 18 locations. The results show that the simulated tsunami arrives at the sensor locations 10 minutes before the observed tsunami, which is roughly a 2% error in velocity over the course of 8 hours of travel time. At several stations, there is good agreement between the observed data and the GeoClaw model. The observed and model results differ most significantly in harbors, where the authors mention that the three dimensional wa-

ter flows in and around harbors are expected to be more turbulent, and the bathymetry resolution may not have been fine enough to properly model the situation. The study also conducted preliminary research on the locations not included previously. These locations are in less agreement, citing possible effects of bathymetric features in the far field.

The results of both Arcos (2015) and Gonzalez (2011) are fairly conclusive in the validity of GeoClaw. With the exceptions of when the assumptions of the Shallow Water Equations were violated, or sufficiently accurate bathymetry was not available, GeoClaw performs well.

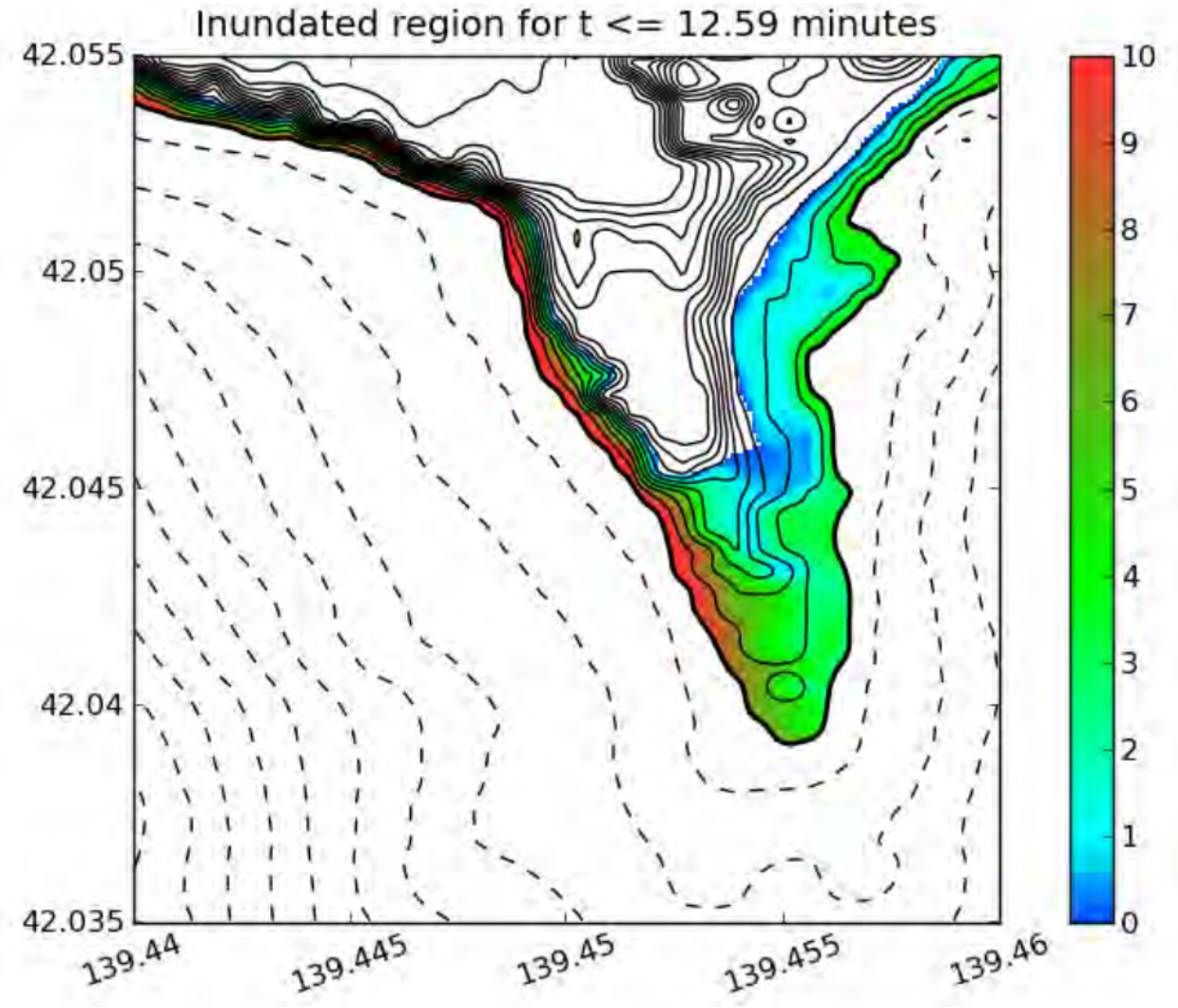


Figure 2.5: The results of a GeoClaw model from Gonzalez et al., 2011, showing the inundation results of a GeoClaw model of the Aonae peninsula on the southern side of the Okushiri Island in Japan. The results of the inundation show a maximum inundation of nearly 10 meters, with waves reaching up to 300 meters inland, which is congruent with literature and field observations.

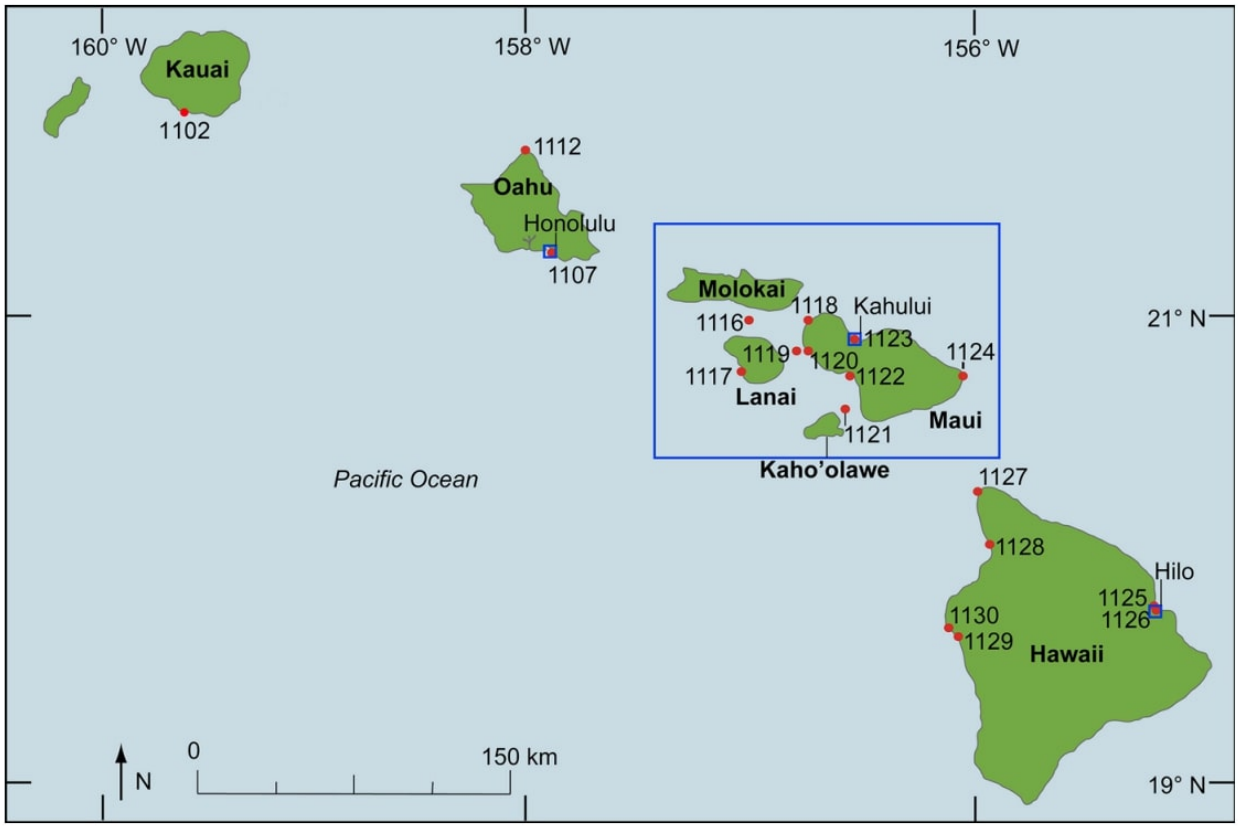


Figure 2.6: The study area used in Arcos (2015), showing the locations of the velocity meters located around Hawaii. For each location labeled, there are several monitors at various depths below, which allows for the capturing of water velocity at several levels in the water column.

2.2 Sensitivity of Tsunamis to Earthquake Fault Parameters

The determination of initial earthquake parameters for a given rupture can take significant time through more traditional means, such as seismic waveform inversion (Johnson, 1998). Tsunami propagation can be sensitive to many of these parameters, as explained in a study by Gica et al., 2007. In this study, a verified tsunami modelling software, Cornell COMCOT, was used to test the sensitivity of tsunamis to each parameter individually, using the same Shallow Water Equations that are utilized by GeoClaw. Under the same magnitude, variations in model source parameters showed that rake, dip angle, epicenter location, and focal depth did not have a significant impact on wave height at different locations, while other factors, including fault dimensions, strike angle, and slip distance did have a profound effect on wave height. The three separate locations tested (Japan, the Aleutian Islands, and Chile) used Hawaii as a target of propagating tsunamis. All three locations had similar affects in the model; the previously specified earthquake parameters had significant affect in wave height, while the others showed little to no impact, as the majority of the sources used seismic waveform inversion to generate a source (Gica, Teng, Liu, Titov, & Zhou, 2007).

A similar method was conducted by MacInnes et al., 2013. Rather than changing the parameters to determine each parameters effects on tsunami propagation, the 2011 Tohoku event was used, comparing several different source models proposed by different research groups with data that was observed from the event. Using GeoClaw, each source was implemented into a model to determine tsunami behavior. Many of the proposed sources were able to replicate observations, and most within a 20% difference from observations. None of the sources, however, were able to replicate the high levels of runup north of 39

degrees North latitude (MacInnes, Gusman, LeVeque, & Tanioka, 2013). The study mentions these results are most likely due to a lack of high-resolution bathymetry, as studies conducted using 50 m resolution was in better agreement with observations (Shimozono et al., 2012). In general, MacInnes et al., 2013 is in good agreement with Gica et al., 2007 in that fault dimensions, strike, and slip distribution are the most impactful source parameters.

A third study, Goda et al., 2014, generated a series of 11 proposed fault geometries that were determined through inversion using different types of data, including teleseismic, tsunami, and geodetic. For each of the 11 fault geometries, 61 models were created through varying strike and dip configurations as well as through the performance of a spectral analysis and synthesis of slip distribution. The results of the study show tsunami wave height is sensitive to dip angle, which is different than the results of Gica et al., 2007. However, the authors cite that the dip of a fault in a subduction zone, where the majority of tsunami causing earthquakes occur, is often deterministically accountable (Goda, Mai, Yasuda, & Mori, 2014), as the dip of a subduction zone tends to be well known, as evidenced by NOAA's SIFT sub-fault system addressed in Section 2.3. Goda et al., 2014 also determined that tsunami wave height has a very high sensitivity to slip characteristics, both in amount of slip as well as the distribution, as the spatial variability can likely cause changes in wave height.

2.3 Sub-Faults in Tsunami Modelling

Gica et al., 2008 briefly describes tsunami sensitivity to certain parameters and validates the implementation of several sub-faults provided by NOAA[1], and implemented in this

[1]https://nctr.pmel.noaa.gov/index.html

study as well. Several models were conducted using the verified MOST propagation model by varying which of these sub-faults were implemented. These models were then compared to empirical evidence gathered from a series of earthquakes, including the October 4, 1994 Kuril Tsunami, the November 17, 2003 Rat Island Tsunami, and the November 15, 2006 Kuril Tsunami. In general, the results of the models were positive, and show a good correlation to the empirical evidence. However, there were some limitations in how the study was conducted. The sensitivity of tsunamis to fault parameters is poorly explored, and is more thoroughly covered in the previously mentioned studies in Section 2.2, as this was not the focus of the study. Through the implementation of the multiple sub-faults, the variation of the epicenter also required the variation of each of those sub-faults parameters, as a new sub-fault often needed to be selected to model the new epicenter. The authors also did not vary the geometry of fault failure, and instead assumed only a single geometry of a two sub-faults wide, along fault axis. This study also did not vary the slip of a given sub-fault, and all of the faults contained a 1m slip, regardless of distance from the epicenter. This did not appear to have an impact on the tsunami propagation, but it is a factor in the determination of the number of sources required to model an earthquake. The NOAA Center for Tsunami Research currently gives these sub-fault sources with an assumed 1 m slip parameter, which correlates to a 7.5 magnitude earthquake given the 100 km by 50 km size of the sub-fault, so this likely intended. This study had several limitations, but potentially validates the implementation of the sub fault method of tsunami modelling.

The method of implementing sub-faults in tsunami modelling is also reinforced by the SWAN-JRC code, which also implements a series of discrete faults to model a larger fault failure (Annunziato, Ulutas, & Titov, 2009). As the implementation of NOAA's SIFT database's sub-faults is the main focus of this work, SWAN-JRC is not thoroughly ex-

plored, but is mentioned to bring attention to multiple sub-fault systems for earthquake ruptures have been created.

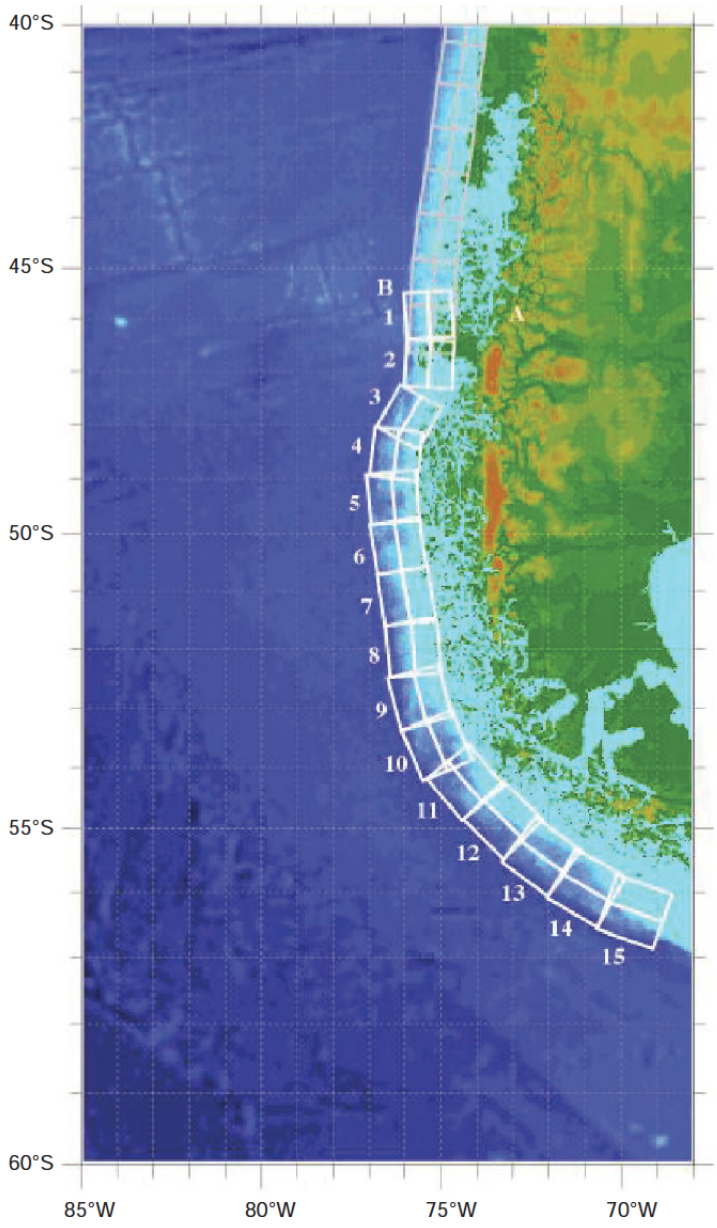


Figure 2.7: The discretization of subduction zone faults in Southern Chile, adapted from Gica, 2008. A series of sub-faults can be used to model a typical subduction zone. The automation of the selection of these sub faults can constrain specific geometries, the impact of which can be shown in Goda et al., 2015. The sub faults described in Section 3.3.1 can be seen in the row labeled 5, with scszb5 being further from the coast and scsza5 being closer. The grayed out section above is the South American Subduction Zone, which has a different name designation and thus is considered a separate subduction zone. Each of these sub faults contain parameters used in earthquake modelling

Chapter 3

Methods

3.1 GeoClaw

As briefly described in the Literature Review section of this paper, GeoClaw is a finite-volume wave propagation numerical modelling code that models geophysical flows, relying on the Shallow Water Equations, shown by 3.1. GeoClaw also implements an adaptive mesh refinement to reduce computation requirements, discussed further in Section 3.1.2. The primary implementation of GeoClaw in this study is for the modelling of co-seismic tsunamis throughout the Pacific Ocean basin. Typically, a user will specify initial conditions of a given earthquake, such as fault parameters, longitude, latitude, and magnitude. The intent behind this study is to automate this process using minimal input from a user, and instead relying on the sub-fault parameters provided by NOAA. The user will give the program a location and earthquake magnitude, and the program will automatically create multiple source files, as described in Section 3.2.1.

GeoClaw utilizes Okada modelling to replicate the displacement of the sea floor (Okada,

1985), which is used as the initial parameters of the tsunami wave propagation. Okada models, in the case of GeoClaw, use inputs to describe a rectangular section of, or a series of, faults. It makes a few assumptions that are specified on Clawpack's website of the Okada model [1]. These include assumptions such as earth is modeled in an elastic half space, and slip is uniform within the rectangular section. The vertical deformation is obtained with a Green's function, with physical properties of the rock assumed to have a Poisson ratio of 0.25 and a shear modulus to be a constant 40 GPa, however these numbers can be changed within GeoClaw. If a longer fault is sectioned into a series of subfaults, the resulting sections would be summed. As the Okada model is linear, the surface deformation will remain independent of the number of pieces that the fault was split into, which can be seen by Figure 3.2, where 56 different subfaults were used to create this model, which are displayed in Figure 3.1.

3.1.1 Shallow Water Equations

The Shallow Water Equations, shown by Equation 3.1, are the governing equations used in GeoClaw. This particular notation is as described by Arcos (2015), where *h(x,y,z)* represents the fluid depth, and *u(x,y,z)* and *v(x,y,z)* are the two dimensional horizontal depth-averaged velocities, with u being eastward movement and v being northward movement. Subscripts denote partial derivatives, and bathymetry is represented by *B(x,y,t)*, and fric-

[1] http://www.clawpack.org/geoclaw/Okada.html

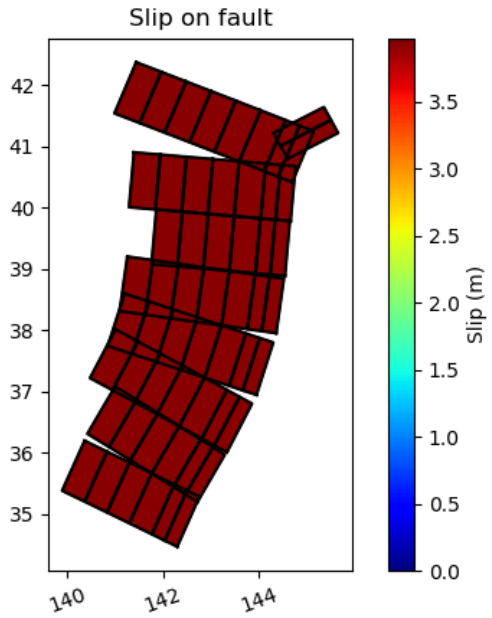


Figure 3.1: An input used for an Okada model, and in turn, a GeoClaw model. Each fault has specific parameters pertaining to location and orientation, as well as a slip distance. It is possible to manipulate the slip distance so that individual faults will vary, but the overall system contains the same amount of energy. In this example, a corrected input that is generated by a python script, described further in Section 3.3.1, where the slip is corrected to conserve energy based on the area, and is thus normalized to 3.975 m.

tional drag by *D(h,u,v)*, which can be determined by equation 3.2

$$h_t + (hu)_x + (hv)_y = 0$$

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x + (huv)_y = -ghB_x - Dhu \tag{3.1}$$

$$(hv)_t + (huv)_x + (hu^2 + \frac{1}{2}gh^2)_y = -ghB_y - Dhv$$

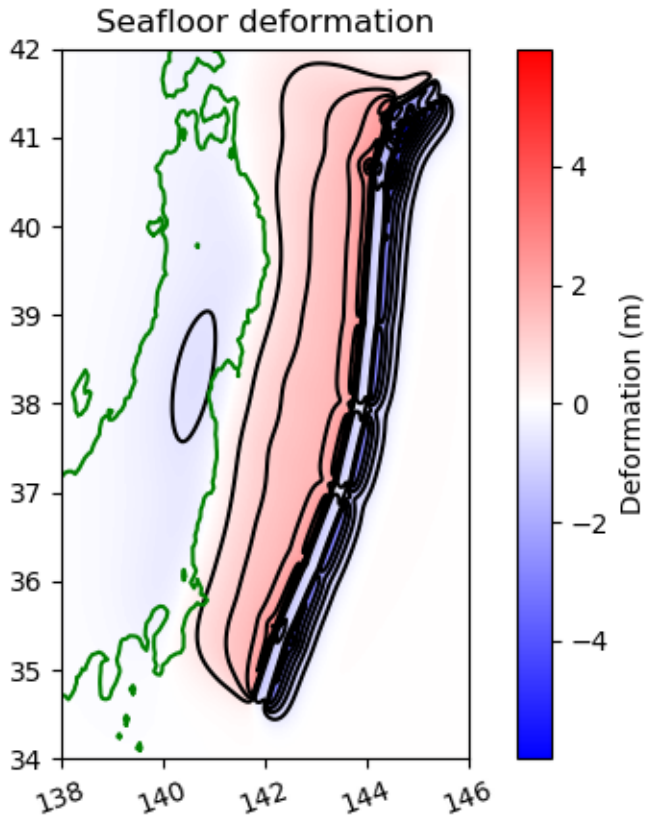


Figure 3.2: An example of an Okada Sea Floor Deformation Model. A deformation to the sea floor surface caused by an earthquake will displace the water above, generating long wavelength tsunami waves. The Tohoku-Oki earthquake, modeled here, can cause a significant area of the sea floor to be displaced several meters, by both uplift and subsiding, resulting in this coupling affect, as can be seen by the red and blue contours, with the coast of Japan shown in green.

$$D(h, u, v) = n^2 g h^{\frac{-4}{3}} \sqrt{u^2 + v^2} \tag{3.2}$$

Where n is the Manning Coefficient. The Manning Coefficient is determined by another equation to model the roughness of a given area of seafloor, and is often estimated, as seen in Arcos (2015), where they varied n = 0.025 to n = 0.035 for smooth and reefed seafloor, respectively, and found little change in the results due to the variation of the Manning Coefficient (Arcos & LeVeque, 2015).

3.1.2 Adaptive Mesh Refinement and Riemann Solver

Adaptive Mesh Refinement is a system used in many computational models to reduce the requirements of computing power to run a model. This is done by increasing and decreasing resolution in areas based on the complexity of the model in that location, and is well explained in Berger & LeVeque (1998). For example, if a tsunami is only a few minutes old in the model, there is no reason to model the opposite side of the ocean basin at maximum resolution, but for best results it is less desirable to decrease the maximum resolution on the side of the basin that the tsunami is generated in. This will minimize the computation requirements of the model being run, but will still deliver good results. This can be seen in Figure 2.3, where the locations around the flood are of the highest resolution (3 meter by 3 meter), especially compared to the west and east most areas of the computational domain, where resolution was lowered to 50 by 50 meter grids.

GeoClaw uses Riemann solvers to handle wave propagation and inundation with the use of adaptive mesh refinement (LeVeque et al., 2011). The concept of Riemann solvers can be seen in Figure 3.3. The shallow water equations are used to determine the values of Q_i

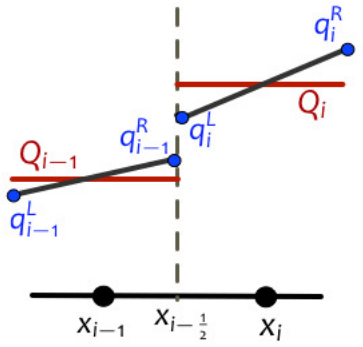


Figure 3.3: An example of the general concept of Riemann solvers used by Clawpack. The mass flux of two adjacent cells can be calculated to conserve the energy of the propagating tsunami wave. Figure from http://www.clawpack.org/riemann.html#d-riemann-solvers

and Q_{i-1}, which are the cell average for the q vector, which is determined by the equation $q = \{h, hu, hv\}$, where h is depth and hu and hv is momentum in the x and y directions, all three of which can be calculated through the Shallow Water Equations (Equation 3.1) and are used as inputs of the Riemann solver.

The solver is the a large portion of the computational requirements of GeoClaw, as it is required at every cell interface and at every time step. GeoClaw's method of modelling starts with the average in a given computational cell, which leads to a series of discontinuities, as can be seen in Figure 3.3. A Riemann solver calculates the mass flux of the cell through a particular cell wall. A multidimensional Riemann solver often simply calculates a one dimensional case in each dimension. This means that for a two dimensional model, the solver needs to be applied to four of the cell walls to determine the cell's total mass flux, while a three dimensional solver will calculate the top and bottom of the cell as well. This allows for the modelling of the the wave propagation to conserve mass and energy through the calculation of the cell's flux.

3.2 Overall Work Flow

The overall work flow of the series of scripts can be best described by Figure 3.4. This shows the order in which the completed project operates, and are described in more detail in the following sections. The program begins by downloading the bathymetry of the entire Pacific Ocean basin, and manipulates it to solve complications with implementing bathymetry across the antimeridian, into an ascii text file. The next script downloads the NOAA SIFT sub-fault database, which contains the parameters of all of the sub-faults needed to vary the geometry of fault failure. Each of these sub-faults contain an area that is 100 km long and 50 km wide. The rupture of these sub-faults with a slip of 1 m will generate a 7.5 magnitude earthquake each.

The next part of the program then generates a series of fault failure geometries based on the inputs of latitude, longitude, and magnitude of an earthquake, which is implemented by the user into a separate Python script to use as an initial location to change earthquake fault parameters. At this point a loop begins, where each geometry is used per iteration of the loop, which continues until all five geometries have been modeled. A geometry, which is a text file containing the selected sub-faults from the SIFT database, is loaded into a script and read. The amount of slip each sub-fault undergoes is corrected to conserve the moment of the earthquake, so that each geometry has the same moment as dictated by the magnitude of earthquake. As the geometries are created, the number of sub-faults are manipulated to correlate to said geometry. If, for example, an along dip geometry is desired with multiple sub-faults, more than are present in an along-dip geometry. To conserve the moment of the earthquake, the sub-faults that are present need to have an increased slip, as changing the area without changing the slip will have an effect on the moment. This is also applied in the instances where a non-integer is required for a given number of sub-faults. The largest integer number of sub-faults is used, and the remaining area not

included is corrected for by changing the amount of slip to preserve the earthquake's moment.

Here GeoClaw will implement these various fault failure geometries, where a change in topography caused by the earthquake releasing energy which deforms the overriding sea floor (referred to as dtopo) is calculated from an Okada model, and placed into a specialized text file with the .tt3 designation. This file is then read by several scripts within GeoClaw to generate the model based on parameters set within the two other main Python scripts in GeoClaw, setrun.py and setplot.py, which determine the computational domain of the model and the plotting domain of the model, respectively. The output files of the completed model are then placed into generic folders in the working directory of the computer, labeled _output and _plots. The contents of these folders are then copied into a new folder within the directory, which is labeled based on the name of the geometry being modeled, and removed from the previous folders. The loop then begins to model the next geometry in the series, until all five geometries created have been modeled.

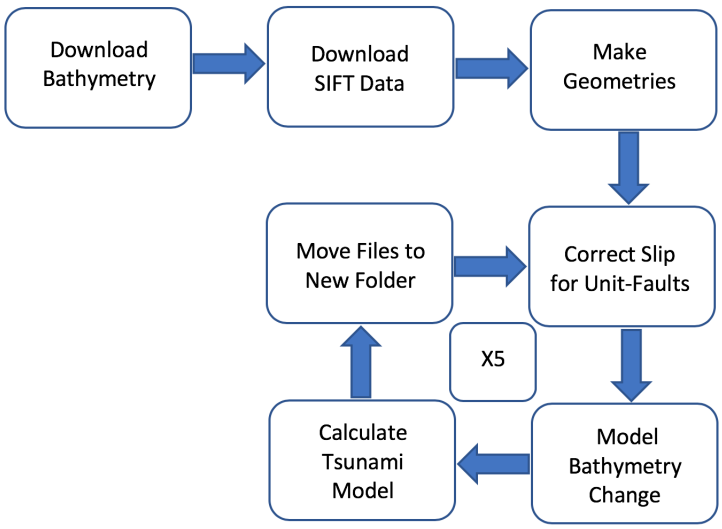


Figure 3.4: This is the general workflow of the overall series of Python programs, which results in the modelling of five separate tsunamis dependent upon potential geometries of sub-fault failure. This figure is described in further detail in Section 3.2

3.3 Program Modifications

This project mostly consists of the changing of GeoClaw programs or the creation of accompanying programs. The modification of the previously existing programs are simple, and will be described in this introduction. The creation of accompanying programs are more significant, and each have their own subsections. The modification to existing scripts was to allow GeoClaw to read a file to determine the sub-fault rather than specifying the fault parameters within the script. This allows the model to implement multiple different inputs for a single run of models, and prevents the user from having to manually implement each sub-fault's parameters. Another modification is to allow GeoClaw to implement sub-fault parameters simply by referring to the sub-fault's name, rather than having to keep to associated information together. Modified GeoClaw programs are available in Appendix 6.

3.3.1 Automatic Fault Geometry Constraining

The automatic fault geometry consists of manipulating either the fault naming system implemented by NOAA (fault details located at `https://sift.pmel.noaa.gov/compressed/info_sz.dat`) or works based on the proximity of a sub-fault to the given epicenter of an earthquake. The names of sub faults from NOAA are in a grid system in the along-strike and along-dip directions, allowing for them to be indexed fairly simply by manipulating the sub-fault's name, as can be seen by Figure 2.7. There are five separate geometries that are determined through this program developed in this work, with four being dependent upon the manipulation of the names of sub faults, and the fifth dependent upon simply finding the nearest number of sub faults that corresponds to the area determined by Equation 3.5. The name manipulation requires the proximity of the closest sub-fault to begin with, but then simply creates desired geometries by changing two aspects of the name. The process of locating the closest sub fault is described by Haversine Equation, as described by Equation 3.5.

$$\begin{aligned} a &= sin^2(\phi_b - \phi_{a/2}) + cos(\phi_a) * cos(\phi_b) * sin^2(\lambda_b - \lambda_{a/2}) \\ c &= 2 * atan(\sqrt{a}, \sqrt{1-a}) \\ d &= R * c \end{aligned} \tag{3.3}$$

Where ϕ is the latitude of a given location, then converted to radians by multiplying by $\pi/180$, and λ is the longitude of one of the desired points converted to radians as well. The location of the epicenter of the desired earthquake is used as ϕ_a and λ_a, and is compared to an array containing every sub fault's latitude and longitude, corresponding to

ϕ_b and λ_b. These values are placed into an array and is then sorted by smallest distance, with the closest sub fault at the top. This sub fault's name designation is then manipulated to vary the geometry into four separate geometries, described in the next paragraph. A fifth geometry is created not by manipulating the sub fault names, but by using the n closest sub faults from the sorted array, where n is determined through Equations 3.5, as described by (Kanamori, 1978) and (Hanks & Kanamori, 1979). These equations dictate the amount of slip a specific earthquake will have, given a specified magnitude and area. As each sub fault described has a preordained area and slip, it is possible to determine the number of sub faults required to make an earthquake of sufficient magnitude.

$$\begin{aligned} M_w &= \frac{2}{3}[(log_{10}(M_o) - 16.1] \\ Area &= \frac{M_o}{(Rigidity \times Slip)} \end{aligned} \tag{3.4}$$

Where M_w is a given earthquake's magnitude, M_o is the earthquakes moment, rigidity is assumed to be 4.3×10^{10} Pascals, and slip is specified to be 1 meter by the sub source values given by NOAA. This slip is varied, however, to overcome limitations of the sub-fault system in that is is increased to correct for an inadequate number of sources. For example, if 2.4 sub-faults are required for a specific magnitude, two sub-faults are used with an increased slip to maintain the magnitude of the earthquake.

After implementing the Equation 3.5, it is possible to then manipulate the name of a sub-fault to find specific sub-faults corresponding to the desired forced geometry. The sub faults all have a name based on a system of naming, which consists of a four letter abbreviation of the subduction zone it is a part of, a letter corresponding to it's position along dip, and a number corresponding to it's position along strike. For example, the fifth

sub fault along the strike and in the second row in the Chilean Southern Subduction Zone would have a name of *csszb5*. When the first sub fault is located using Equation 3.5, the letter and number of the sub fault can be manipulated to locate corresponding sub faults for a desired geometry of fault failure. An earthquake with a magnitude of 8.8, such as the Chilean 2010 earthquake, would require 92.4 sub faults that are 100 km long and 50 km wide to model. If we were to model an along-strike geometry, 92 sub-faults would be divided by two, and subtracted from the number designation in the sub faults name. That sub fault would then be recorded into an array, and the next n faults would be appended into that array. A similar method is conducted for two rows along-strike, except the number of sub faults is divided by four, and the letter of where the second sub fault along-depth is determined by utilizing Equation 3.5 to find the second closest sub fault in the same row as the closest. This way, the two rows will straddle the epicenter given. In the case of an along-dip geometry, a similar process of name manipulation is conducted where the whole variety of letter combinations of along-dip designations are appended in order of closest sub faults until the maximum number is reached. In the geometry of a pair of along-dip sub faults are determined, the closest row of along-dip sub faults that is in a separate row from our first sub fault is used, until the maximum number is reached.

In cases where the number of sub faults required exceeds the number of sub faults available, the slip distance is adjusted to correspond with the same magnitude earthquake. For example, in the case of *scszb5* and the along-dip geometry, there are only sub faults in that row, as seen by Figure 2.3: *scsza5* and *scszb5*. In this case, the slip would be changed from 1 m to 46.3 m. This slip distance is significantly high, and unlikely to occur at this magnitude, as the 1960 Valdivia earthquake's slip motion, which is one of the most powerful earthquakes to ever be recorded at a magnitude of 9.5, is estimated to be around 40 m in select locations (Barrientos & Ward, 1990). This would also be applied to the

others to correct for the need of a percentage of a sub fault, as it is not possible to implement an incomplete sub-fault easily. The entire script to manipulate sub-fault geometries can be found in Appendix 6.

3.3.2 Bathymetry Across the Antimeridian

A complication that arose was for the need to use a large bathymetry file which spanned the entire Pacific Ocean Basin to implement in GeoClaw. Ordinarily this would not be a complication, as a bathymetry file for a specific study area can easily be obtained through NOAA's website[2]. There are several methods that can be implemented to do so, such as using an interactive map, or specifying a window of data through coordinates. There are also several different data sets available for download, such as ETOPO1 Ice, ETOPO1 Bedrock, Coastal Relief Models, etc. The difficulty comes from downloading this data across the anti-meridian. At this location, the latitude and longitude coordinate convention changes from -180 to 180 degrees. This sudden difference in coordinates is difficult to resolve by downloading NOAA's data without manipulating the bathymetry files, which were required by this study.

A program was created to automatically download bathymetry data from NOAA (as seen in Appendix 6), using a range from 120^o E to 310^o E and from 60^o S to 60^o N in two separate files. Both files overlapped at the anti-meridian, with one file on the eastern side and the other on the western. Both of these files are loaded into two arrays, with one of the files having the anti-meridian omitted as to not duplicate that column of data, and were manipulated to create a single data set that spanned the anti-meridian but did not use coordinates to correlate to the bathymetry data. Instead, the location in the array and the interval distance between points in the array are used to determine the latitude and

[2]https://www.ngdc.noaa.gov/mgg/global/

longitude of a sea floor height value. The header information, including the bottom left corner coordinates of the data as well as the spacing of each point, needed to be rewritten to now correlate with the new bathymetry size, as the length of values in the array need to be read by GeoClaw before it begins to read a second line. This allows GeoClaw to read the new data set as a singular set of bathymetry and not need to use the coordinate issues associated with multiple data sets.

Chapter 4

Results

4.1 Wave Gauges

For both sets of case studies, the state of Hawaii was used as the tsunami destination. 18 artificial wave gauges were placed around the islands, as seen by Figure 4.1. Figures 4.2 through 4.11 are zoomed in on the individual gauges. Many are located at the mouth of harbors, with some in channels between the islands. Some of the locations, due to the resolutions used in the modelling, were calculated to be on land. This could be corrected by simply moving the locations or increasing resolution, although these locations were selected as they are the same locations used in Arcos & LeVeque, 2015, and the resolution was what was publicly available from NOAA.

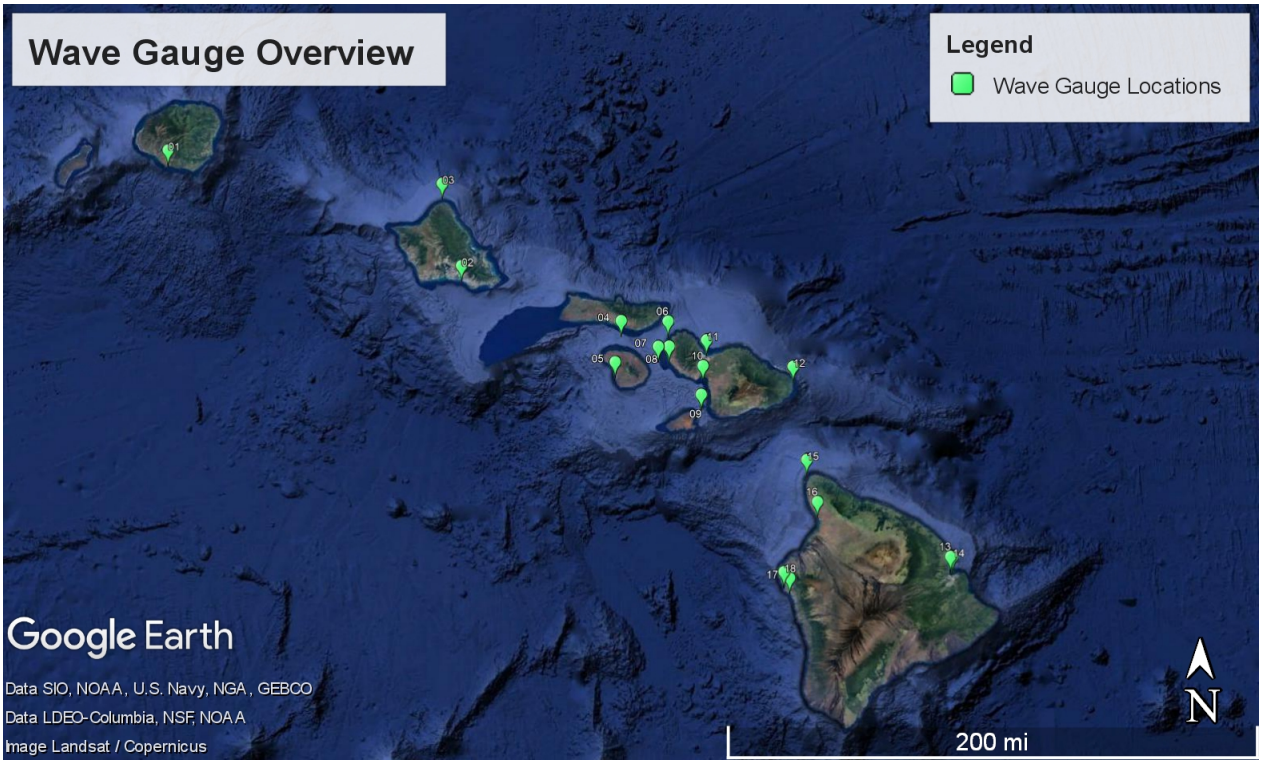


Figure 4.1: An overview of the artificial wave gauges implemented by GeoClaw in a Google Earth file, with the gauge locations shown with green markers. Each gauge is capable of recording mass flux in the x, y, and z directions as well as the wave height at each time step.

Figure 4.2: Gauge 01 is located in Hanapepe Harbor, HI.

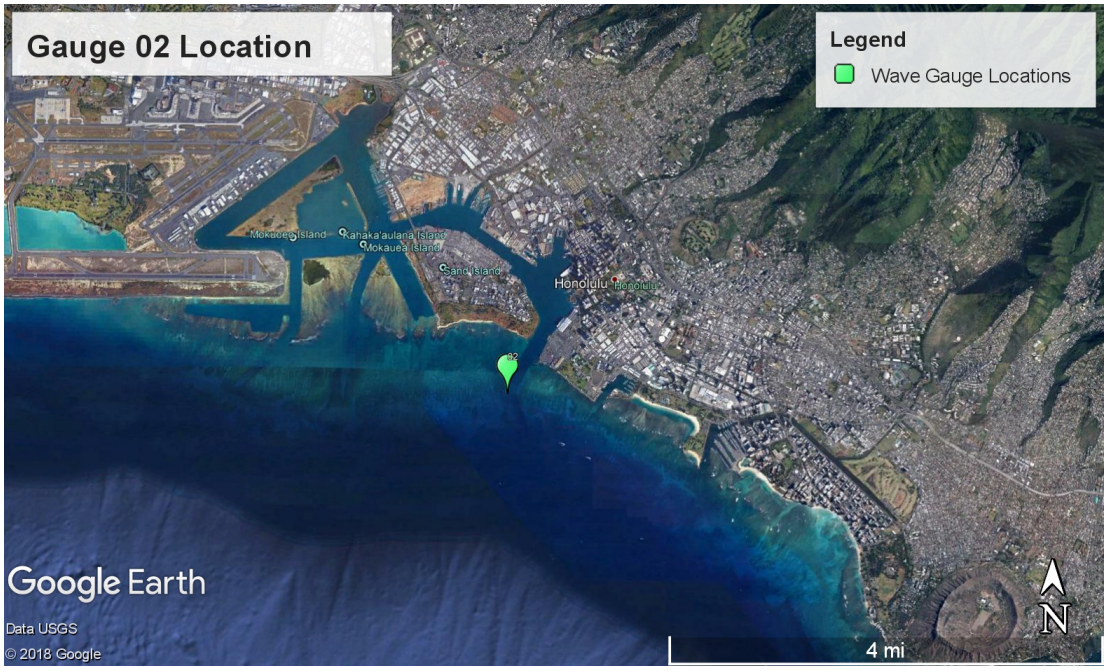


Figure 4.3: Gauge 02 is located in Honolulu Harbor, HI.

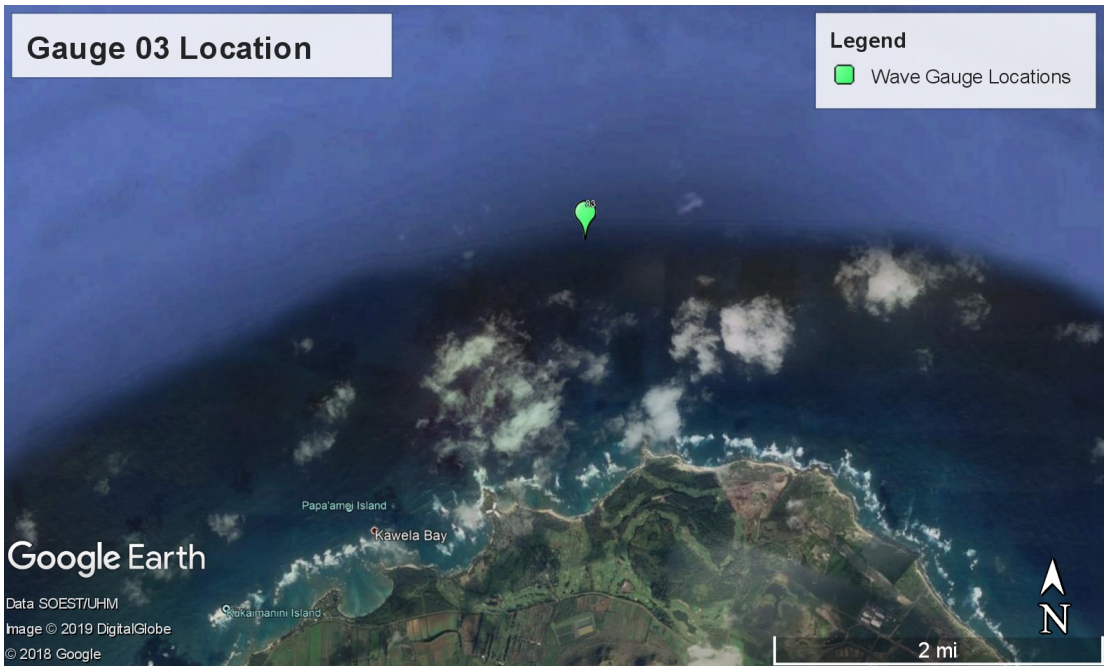


Figure 4.4: Gauge 03 is located North West of Kawela Bay, HI.

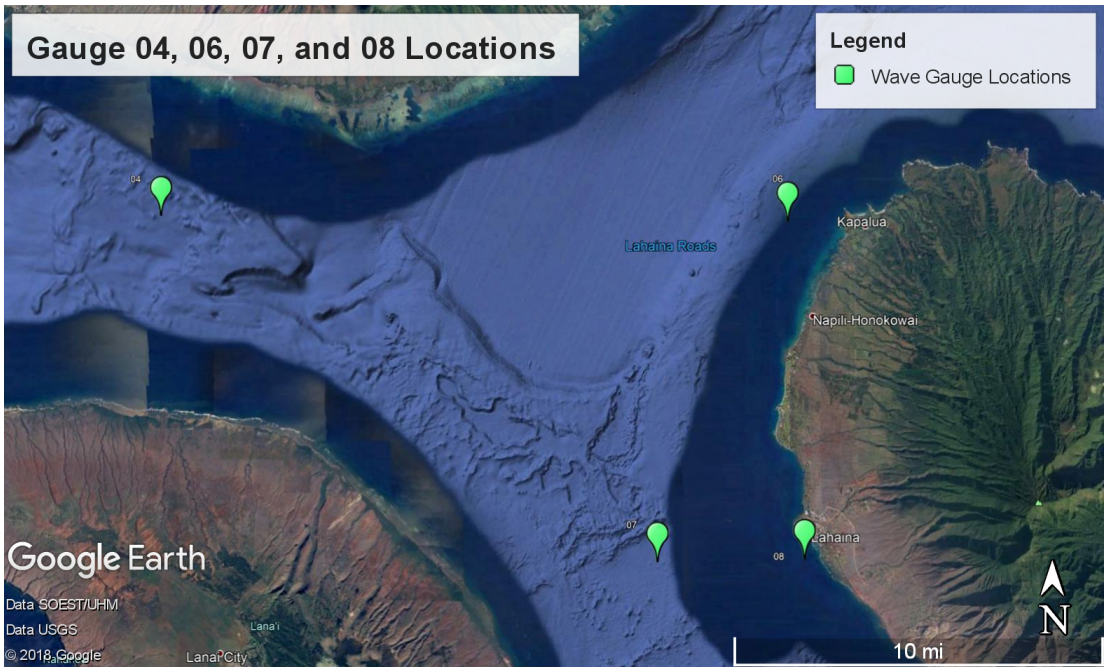


Figure 4.5: Gauge 04 is located between Moloka'i and Lana'i Islands. Gauge 06 is South West of Lana'i Island, and 07 is between the islands of Maui and Lana'i. Gauge 08 is South West of Lahaina.

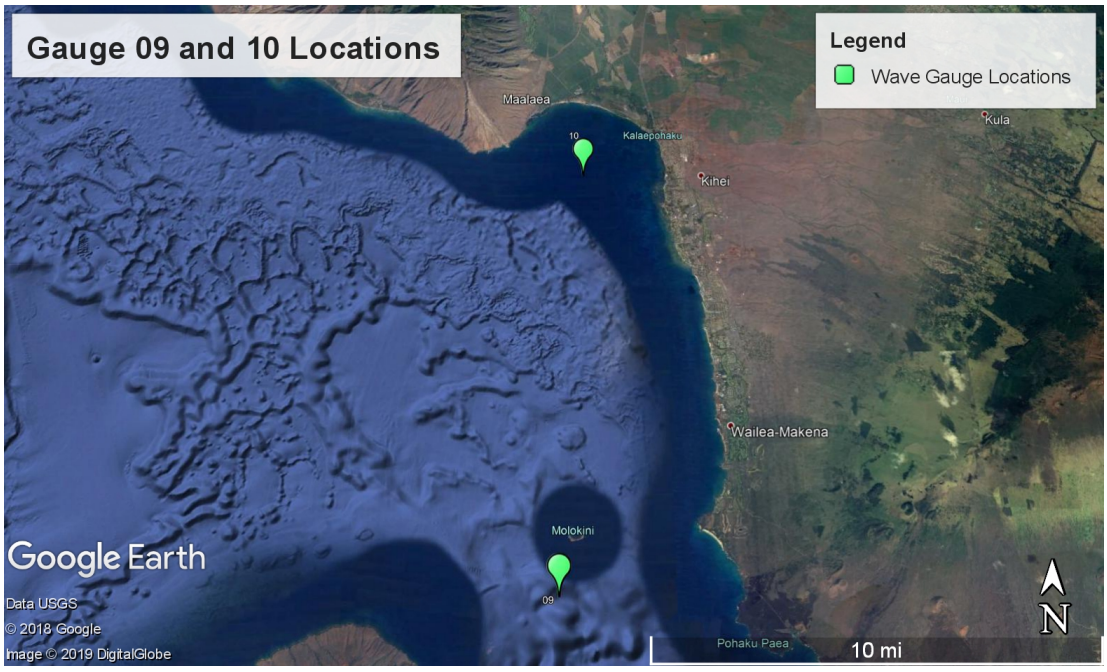


Figure 4.6: Gauge 09 is South West of Molokini, while Gauge 10 is South East of Maalaea. Gauge 11 is North of Kahului, HI.

Figure 4.7: Gauge 11 is located North of Kahului Bay, HI, just outside of the harbor jetties.

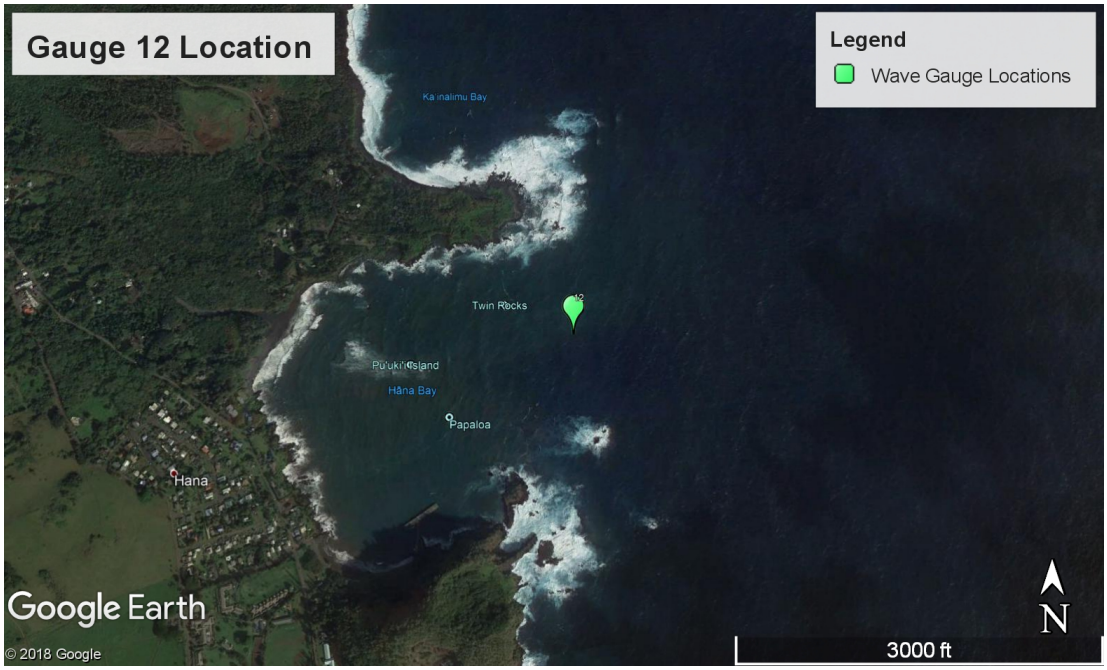


Figure 4.8: Gauge 12 is located in Hana Bay, HI..

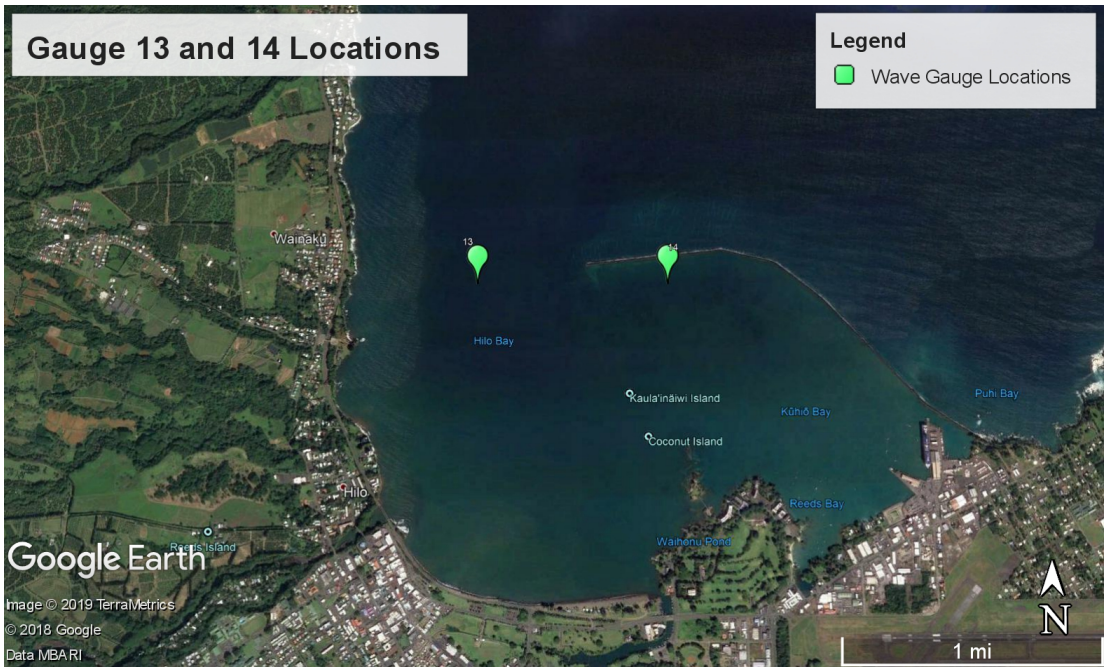


Figure 4.9: Gauge 13 is located in Hilo Bay, and Gauge 14 is also in Hilo Bay, behind the breakwater.

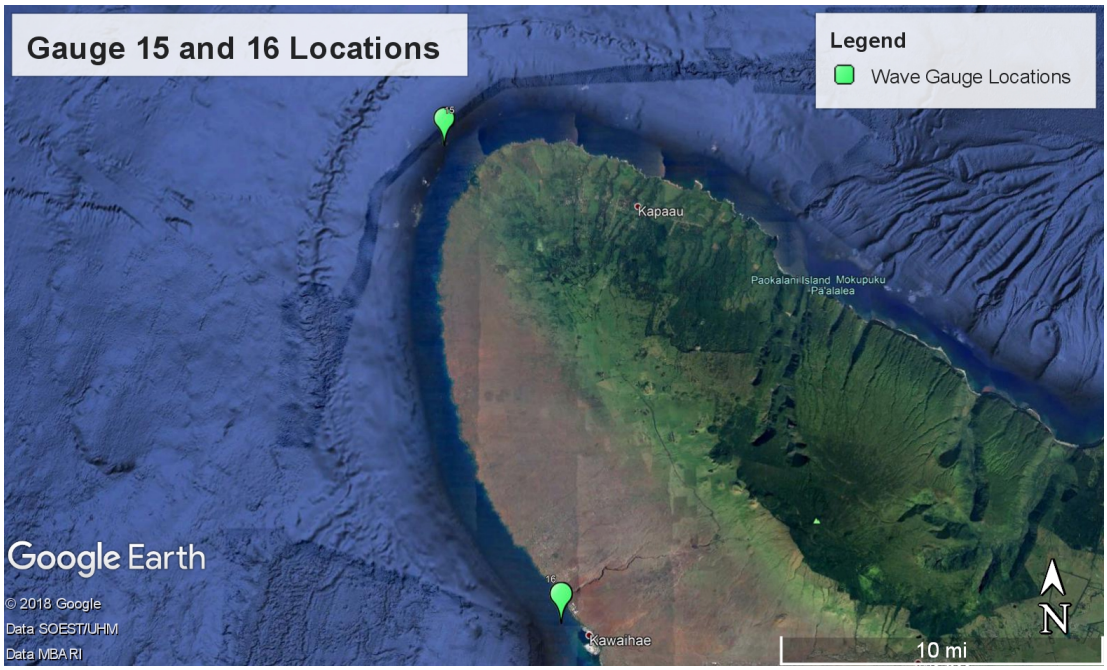


Figure 4.10: Gauge 15 is located North West of Kapaau, and Gauge 16 is located Kawaihae.

Figure 4.11: Gauge 17 is located in Kailua-Kona Harbor, and Gauge 18 is located just North West of Kailua-Kona.

4.2 Chilean 2010 Earthquake Case Study

At 3:30 am local time on February 27th, 2010, a magnitude 8.8 earthquake occurred off the coast of Chile. The epicenter is located at 35.909 ^{o}S, 72.733 ^{o}W, approximately 1.9 miles off the coast (Duputel, Rivera, Kanamori, & Hayes, 2012). This earthquake caused 550 km of fault line to rupture, and along with the resulting 29 m tsunami, caused over 500 deaths (Yamazaki & Cheung, 2011). This also resulted in over $30 Billion USD in damages, impacting over 12 million people (Siembieda, Johnson, & Franco, 2012). Chile, located in South America, is also located near the subduction zone where the Nazca plate is subducting under the South American plate, making this area prone to both many small earthquakes and periodic large earthquakes, as seen by the Mw 8.5 in 1922, Mw 9.5 earthquake in 1960, and Mw 8.0 in 1985 (Hayes et al., 2017). This earthquake and resulting tsunami is an often used example in GeoClaw, and is the pre-set parameters that are in the example script when GeoClaw is downloaded. Because of this, the Chilean 2010 earthquake is used as a case study to observe the effects of the automatic fault geometry constraining explained in Section 3.2.1.

4.2.1 Chilean Modelling Results

The automatic fault geometries are described earlier in Section 3.2.1, the parameters of which can each be seen in Tables 6 through 6. Figure 4.13 is a GeoClaw tsunami model with a single row of along-dip faults within the South American Subduction Zone described by Gica et al., 2008 as the input of the model Figure 4.13a. The sea floor is deformed, as described by an Okada model, seen in Figure 4.13b, displacing the water above. A model time of 0.08 hours, or 4.8 minutes, exemplifies the difference between the other fault failure geometries' impacts on the short term affects of wave displacement. In

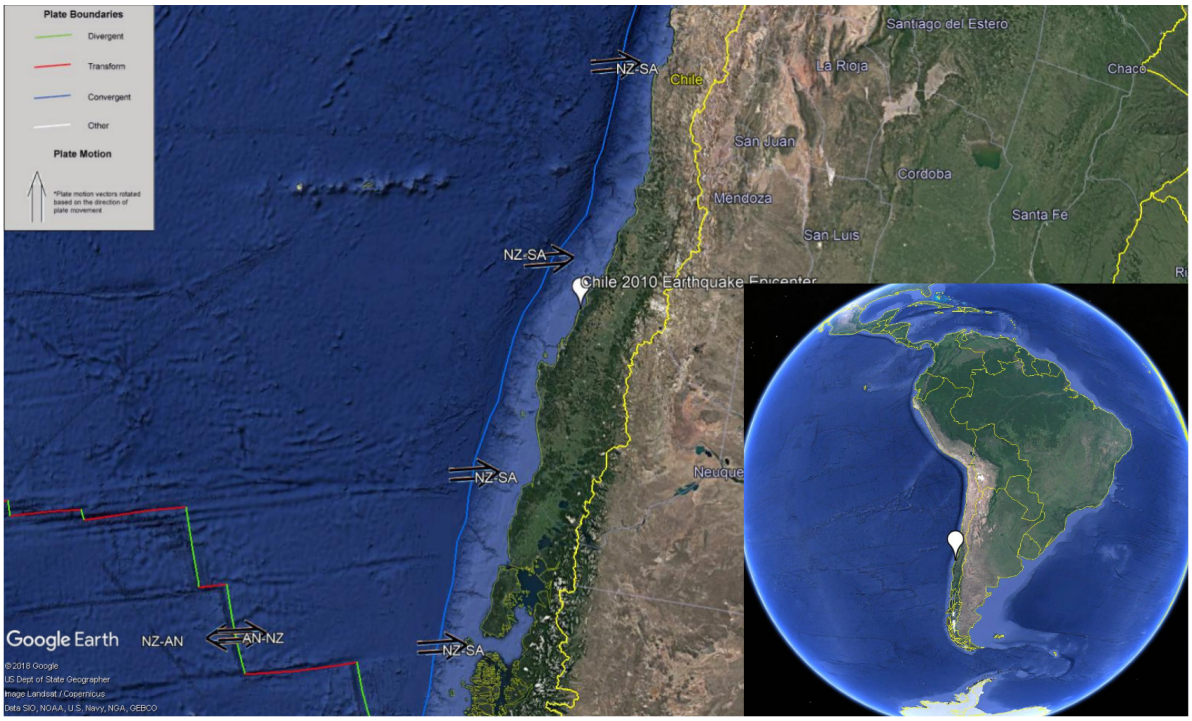


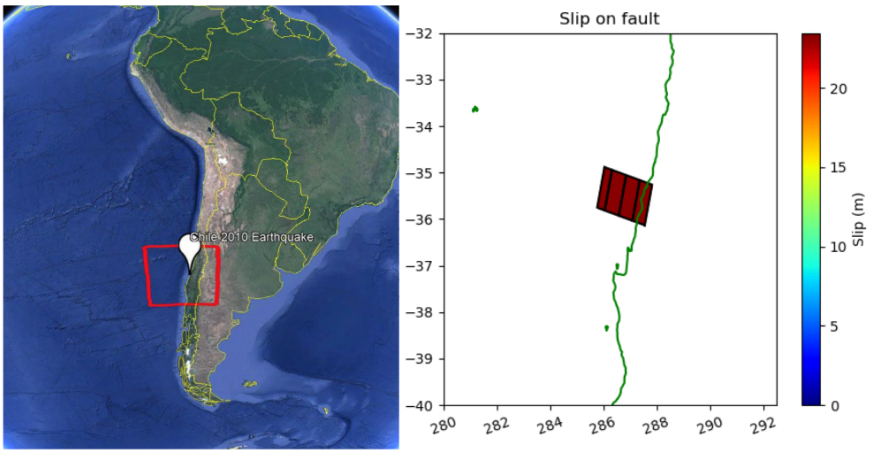
Figure 4.12: The epicenter of the Chilean 2010 earthquake from a Google Earth .KMZ file, along with an overlay from the USGS showing tectonic plate interactions, where the Nazca plate can be seen subducting under the South American plate, creating the subduction zone where large earthquakes are prone to occur. This is the location input of the geometry constraining python script, with a location of 35.909^o S, 72.733^o W, located in the Southern Chilean Subduction Zone, described in Gica et al., 2008.

this failure geometry, it appears that there is primarily a raising of the water surface elevation with little to no depreciation in the normal water surface. Conversely, Figure 4.14, a failure geometry of two along-dip sub-faults, which is represented by the Okada model in Figure 4.14b, and fault geometry in Figure 4.14a, shows depreciation of the water elevation to the southwest of the main swell. Figures 4.19a and 4.19b are geometries that are constrained to the along strike direction in a single and double row, respectively. Each contain alternating areas where the water column raises and lowers, caused by the deformation of the Okada models seen in Figures 4.15b and 4.16b. Lastly, Figure 4.20a is the 'unconstrained geometry, where simply the nearest number of sub-faults are located.

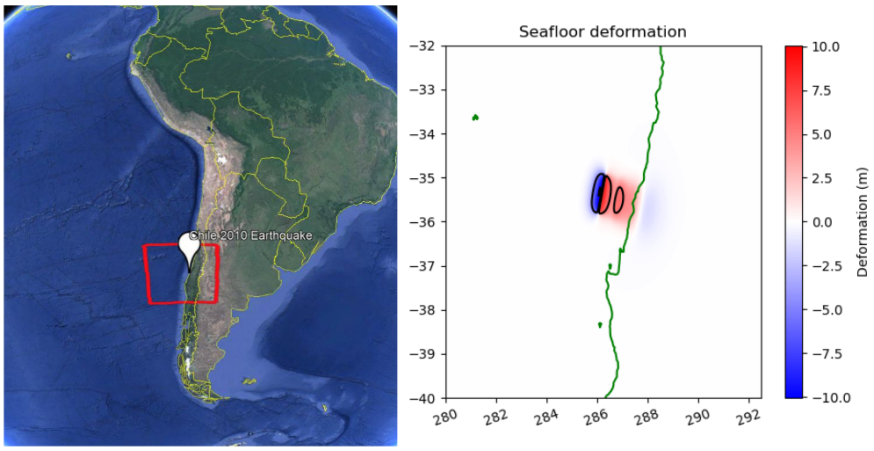
This figure shows that there is a significant displacement in both the positive and negative directions of the water column, caused by the displacement of the sea floor as seen in the Okada model Figure 4.17b. Each model has a resolution of 1 arc-minute at the coarsest, and can be refined for higher resolutions, as dictated by the user. In the case of Figures 4.18 through 4.20, the resolution was forced to maximum in GeoClaw around the study area for a clearer understanding of the water column behavior soon after the earthquake rupture.

Both Figures 4.18a and 4.18b are drastically different in appearance from Figures 4.19a, 4.19b, and 4.20a, and are different in their propagation behavior. Figures 4.19a, 4.19b, and 4.20a contain a significant number of locations where the water level lowers significantly, greater than 0.20 m, while Figures 4.18a and 4.18b have either little or no water level depreciation. Figure 4.19a is represented by a single row of along-strike sub faults through the South American Subduction Zone, whose fault geometry can be seen by Figure 4.13a, and the Okada model in Figure 4.13b. Similar in behavior to Figure 4.19a, Figure 4.19b is composed of two rows of along-strike sub-faults. The geometry and orientation of this model is displayed in Figure 4.14a, and the sea floor deformation model is shown by Figure 4.14b.

Figure 4.20a is a series of sub-faults in which the geometry is not forced, which can be shown in Figure 4.17a, and the accompanying Okada model in Figure 4.17b. Using Equation 3.5, it is possible to determine the number of sub-faults are required to rupture a given area with a specified amount of slip. This geometry is likely to be the most reasonable, as it simply selects the nearest number of faults required to approximate the area required to rupture. It is also imperfect, as the slip is uniform throughout all of the sub-faults. Both the shortcomings as well as potential improvements are discussed upon in Chapters 5 and 6, respectively.

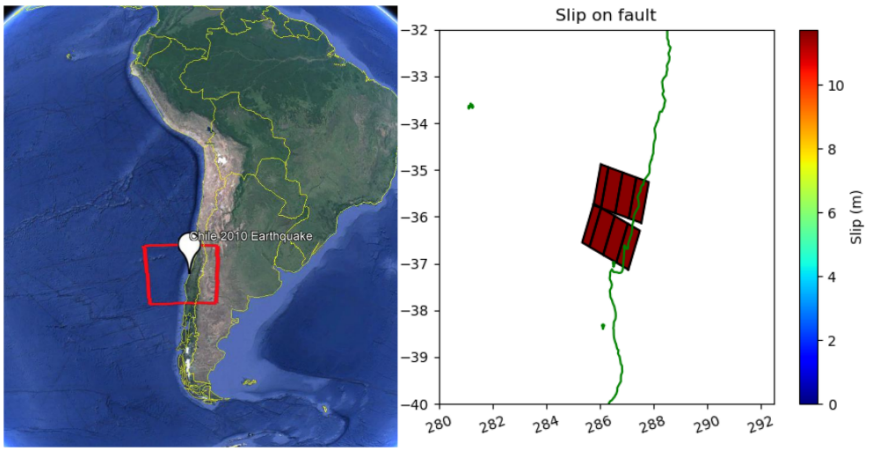


(a)

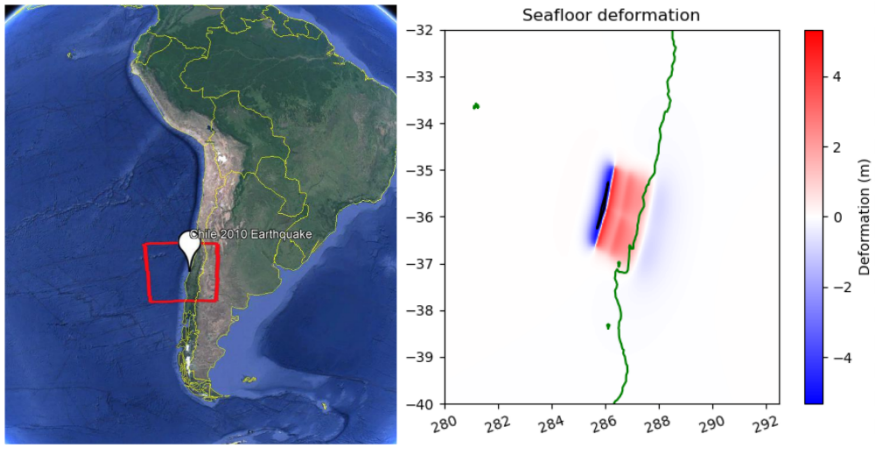


(b)

Figure 4.13: The distribution of sub-faults and deformation from an Okada model of a single row of along-dip geometry, with the model having a 5 m contour interval, shown by sub-figures a and b respectively. The sea floor deformation has a maximum elevation change of 10m in both uplift and subsidence.

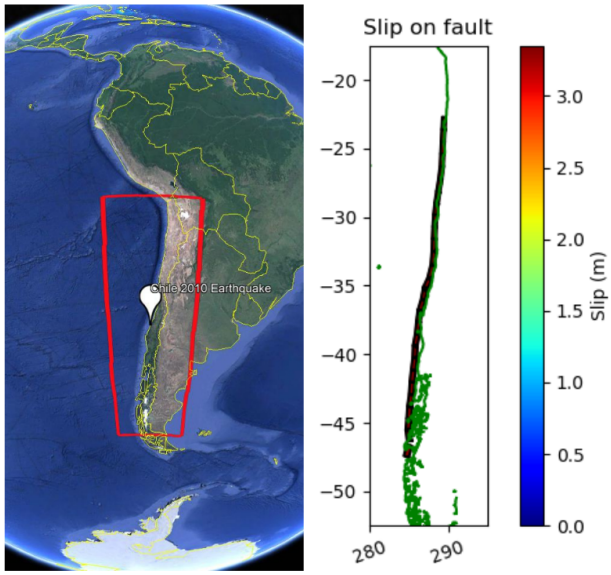


(a)

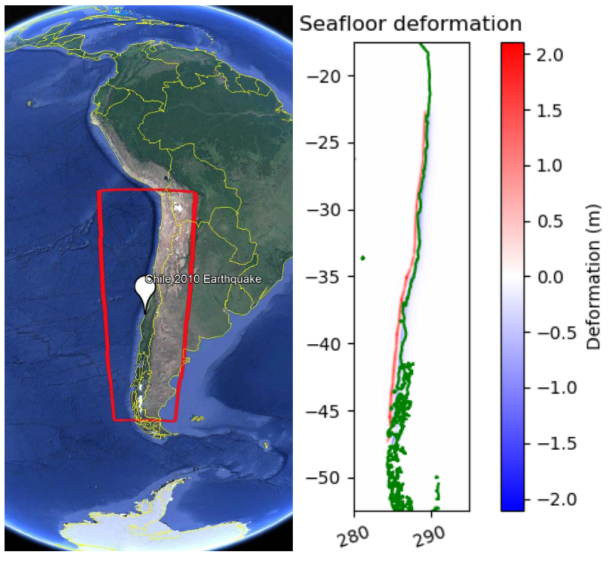


(b)

Figure 4.14: The distribution of sub-faults and deformation from an Okada model with two rows of along-dip geometry, with the model having a 5 m contour interval. Sub-figure a shows the slip distribution, whereas sub-figure b shows the sea floor deformation calculated from the okada model. The sea floor deformation has a maximum elevation change of 4.5 m of both uplift and subsidence.

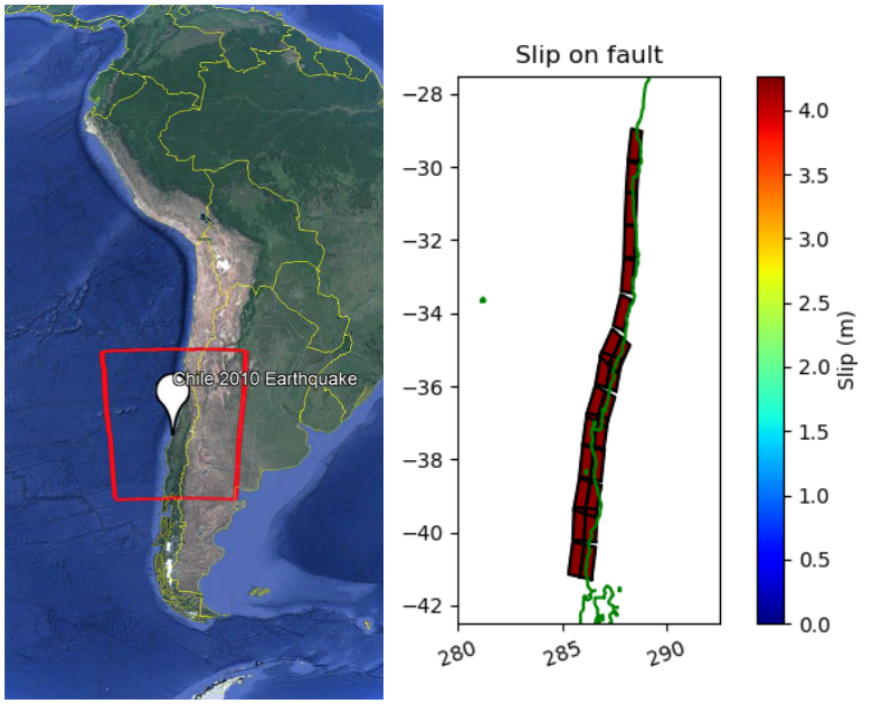


(a)

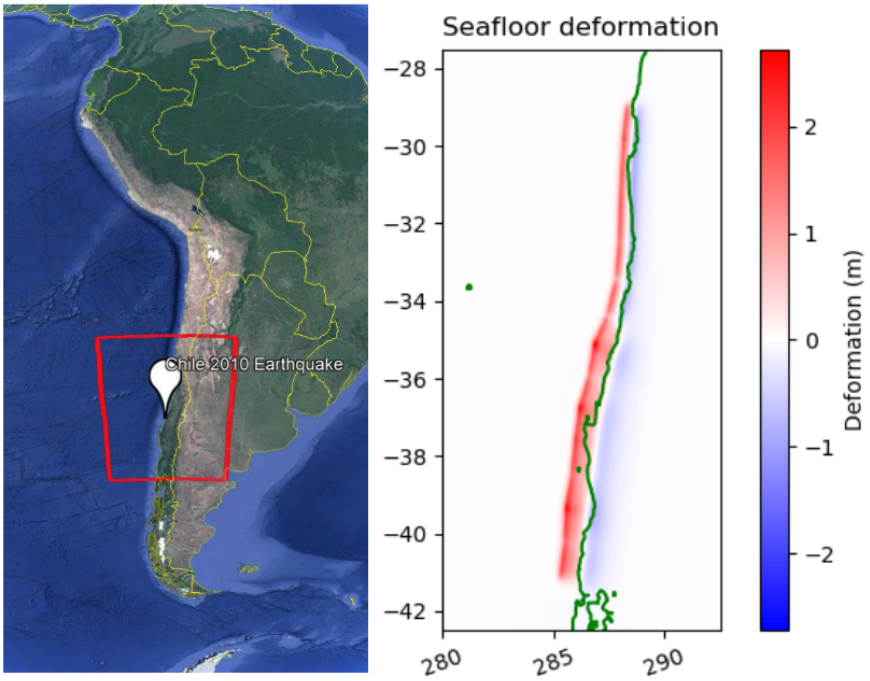


(b)

Figure 4.15: The distribution of sub-faults and deformation from an Okada model of an along-strike geometry, with the model having a 5m contour interval. Sub-figure a shows the slip distribution of the fault failure, while b shows the Okada deformation. The sea floor deformation has a maximum elevation change of 2 m in uplift and, 1.5 m in subsidence.

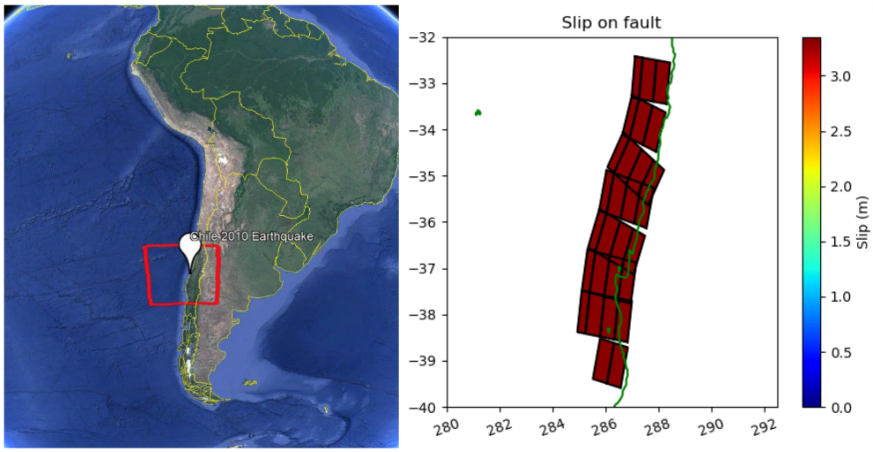


(a)

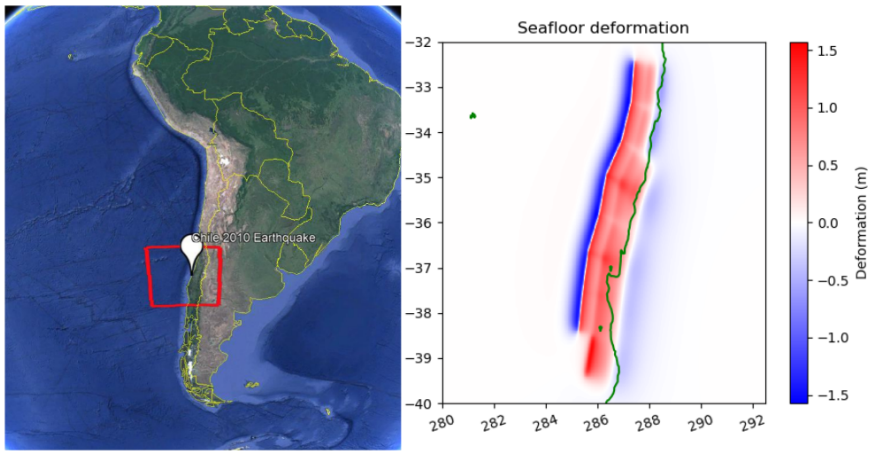


(b)

Figure 4.16: The distribution of sub-faults and deformation from an Okada model of two along-strike geometry, with the model having a 5m contour interval. Subfigure a shows the slip distribution of the fault failure, while b shows the Okada deformation. The sea floor deformation has a maximum elevation change of 2.5 m of uplift, but only 0.5 m of subsidence.

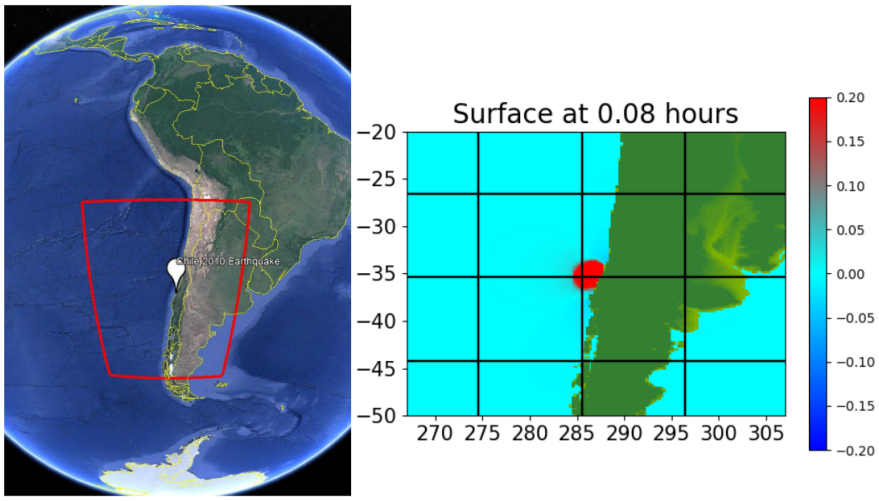


(a)

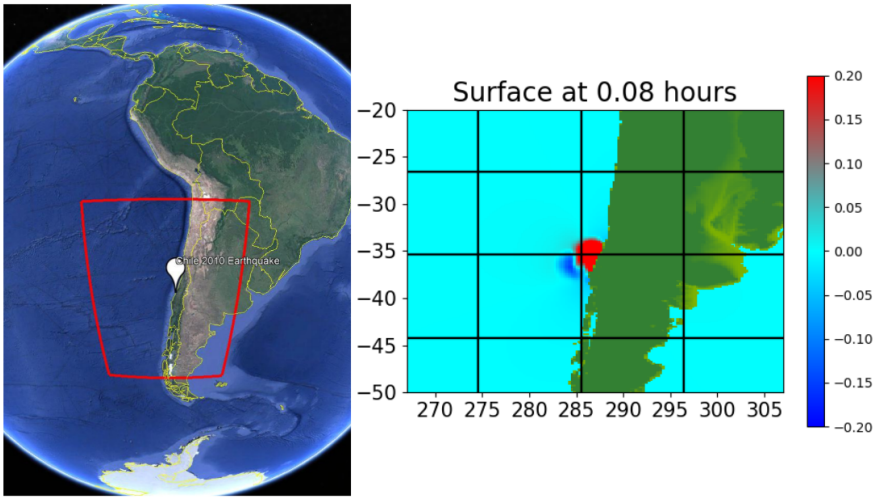


(b)

Figure 4.17: The distribution of sub-faults and deformation from an Okada model with two rows of along-dip geometry, with the model having a 5 m contour interval. Sub-figures a shows the distribution of sub-faults and their associated slip values, while subfigure b shows the sea floor deformation, with a maximum elevation change of 1.5 m in uplift and subsidence.

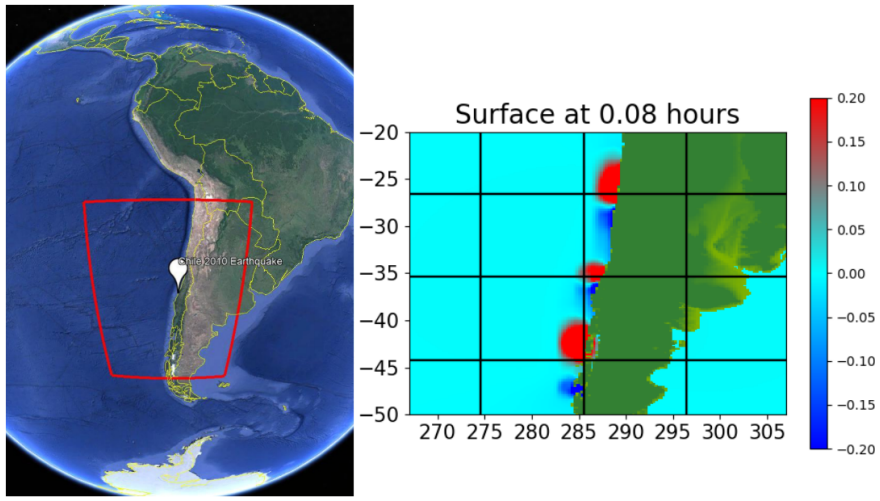


(a)

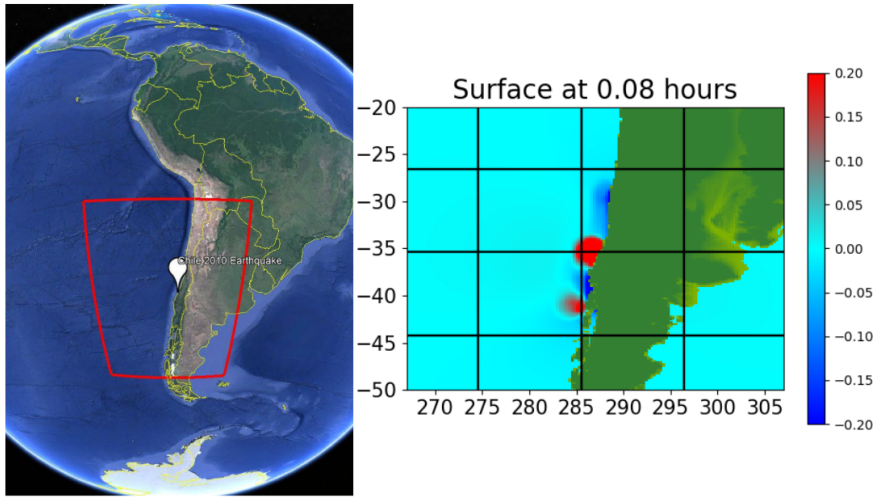


(b)

Figure 4.18: The two along-dip geometries, with sub-figure a showing the single row of sub-faults, and sub-figure b showing two rows of along-dip geometry. The two rows of along-dip geometry shows a large volume of water undergoing sea level lowering, whereas the single row does not show any drop in water level.

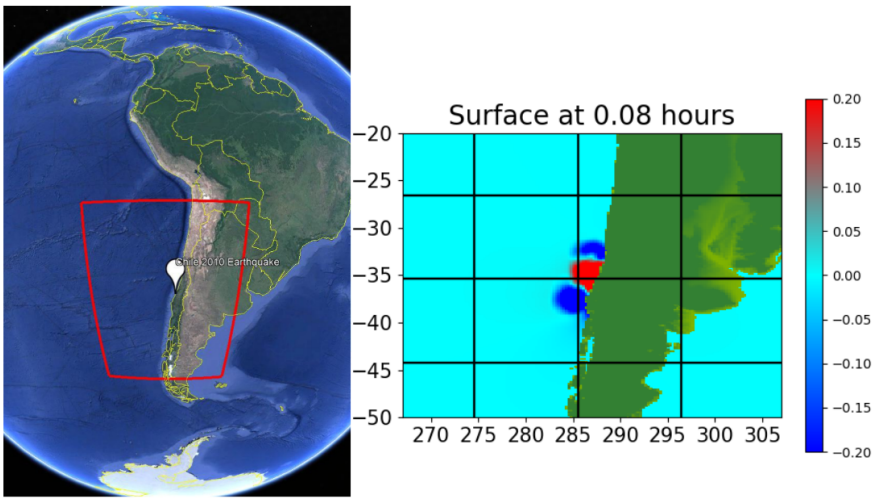


(a)



(b)

Figure 4.19: The two along-strike geometries of the Chilean 2010 earthquake. Sub-figure a shows both water level raising and lowering along the majority of the western coast of South America, where as sub-figure b shows a more concentrated area of water level change that is not as extensive.



(a)

Figure 4.20: The water elevation change due to the unconstrained geometry determined by the Haversine Equation. It shows concentrated areas of water level lowering both in the north and south, with the central area showing a large amounts of water level raising

4.2.2 Wave Gauges for the Chilean 2010 Tsunami

As stated in Section 4.3.2, the wave heights for each model were recorded at 18 wave gauge locations, recording the mass flux in the x, y, and z directions as well as wave height. The wave heights are displayed below, in Figures 4.21 through 4.32. Some gauges, such as 04, 07, 08, 11, 12, 15, and 16, as seen in Section 4.3.2 again, showed a consistent elevation, caused by the adaptive mesh refinement and resolution causing the locations that these wave gauges are located at to be on land. This could be overcome with moving the locations, but these particular locations, as stated previously, were selected as they were the same wave gauges that are implemented in Arcos & LeVeque, 2015.

As can be seen in the following figures, each distinct geometry had a significantly different wave pattern. In general, the largest waves were caused by the two sub-fault wide, along-dip geometry, with an average wave height of less than 0.00 5m. Compared to what was seen, with wave heights ranging from 0.12 m to 0.98 m, the wave height calculated by

GeoClaw was significantly smaller by up to two orders of magnitude.

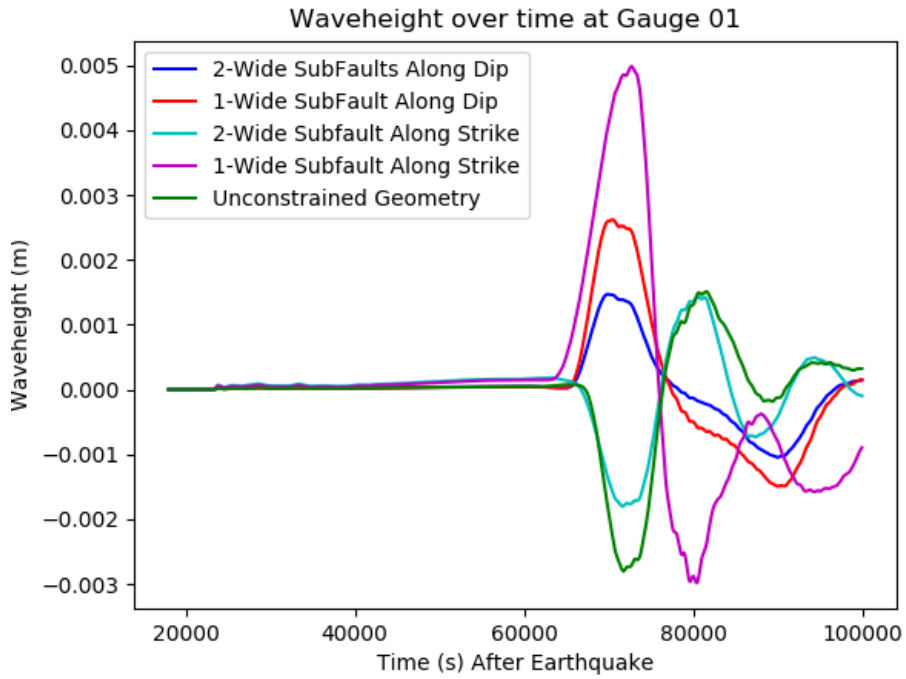


Figure 4.21: Wave height over time for the five different scenarios at Gauge 01 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak, and a -0.003 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The second lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0023 m.

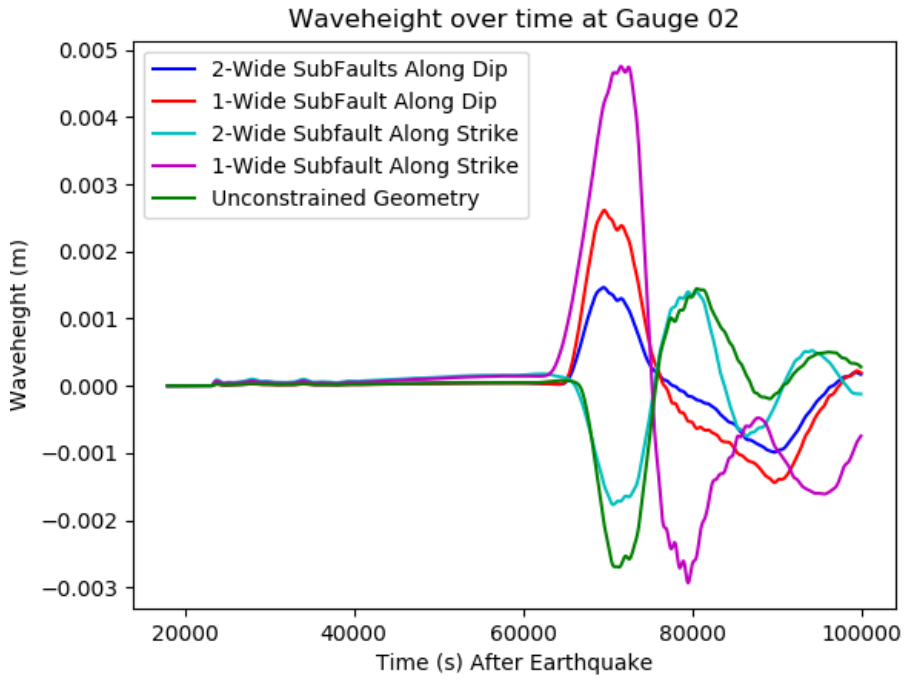


Figure 4.22: Wave height over time for the five different scenarios at Gauge 02 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.0048 m peak, and a -0.003 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The second lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0028 m, as seen at Gauge 01.

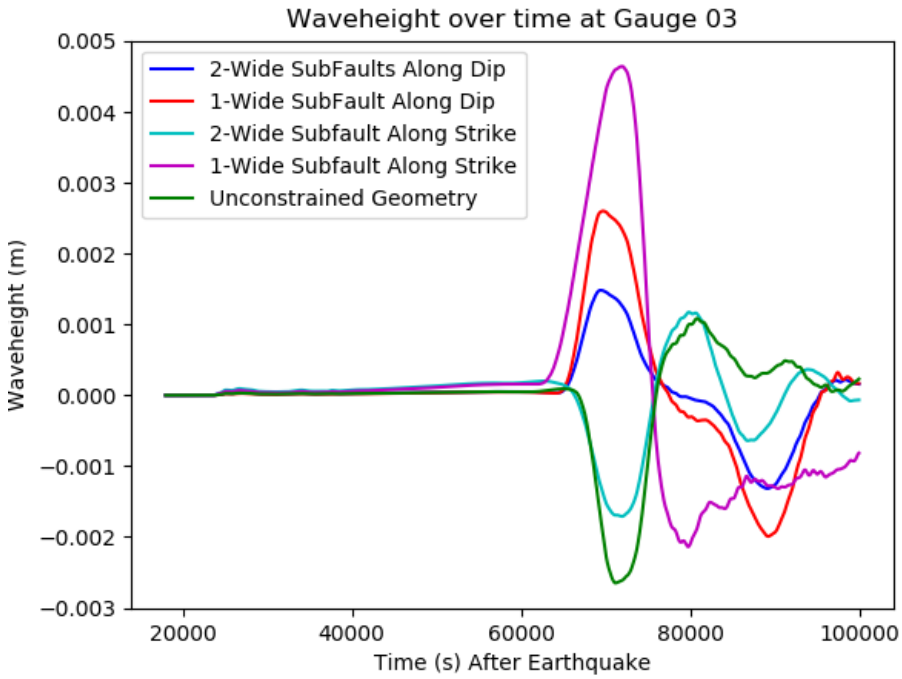


Figure 4.23: Wave height over time for the five different scenarios at Gauge 03 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.0048 m peak, and a -0.002 m trough, although this geometry has the second lowest trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough was caused by the unconstrained geometry of sub-faults at -0.0028 m..

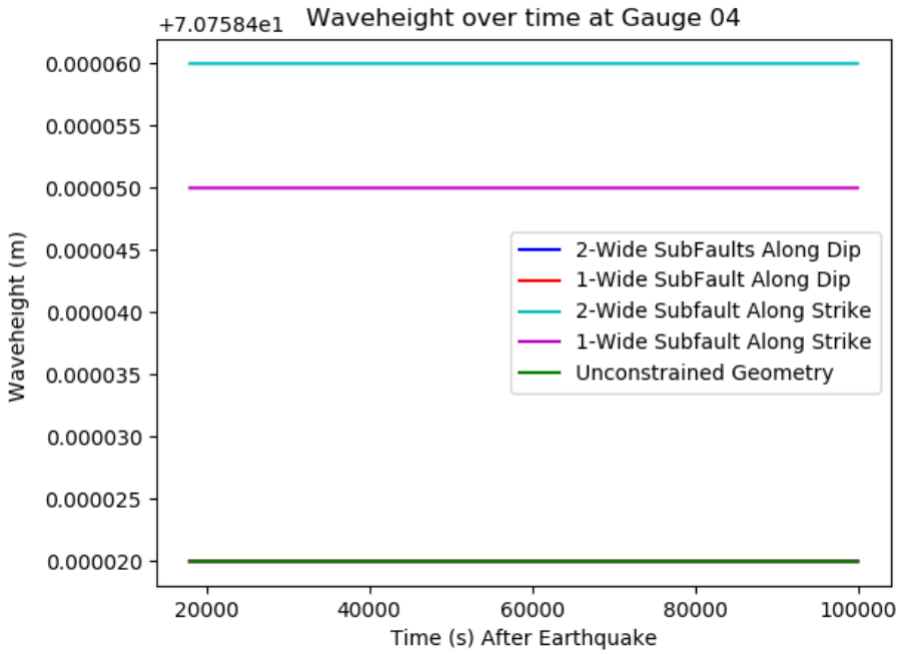


Figure 4.24: Wave height over time for the five different scenarios at Gauge 04 during the Chilean event. The adaptive mesh refinement and grid system used in GeoClaw calculated the cell this gauge was located in as land. Thus, each geometry shows a constant elevation, and in this case, an elevation of just over 70m. It is unclear why each geometry had a different constant elevation, although variations are in the mm range.

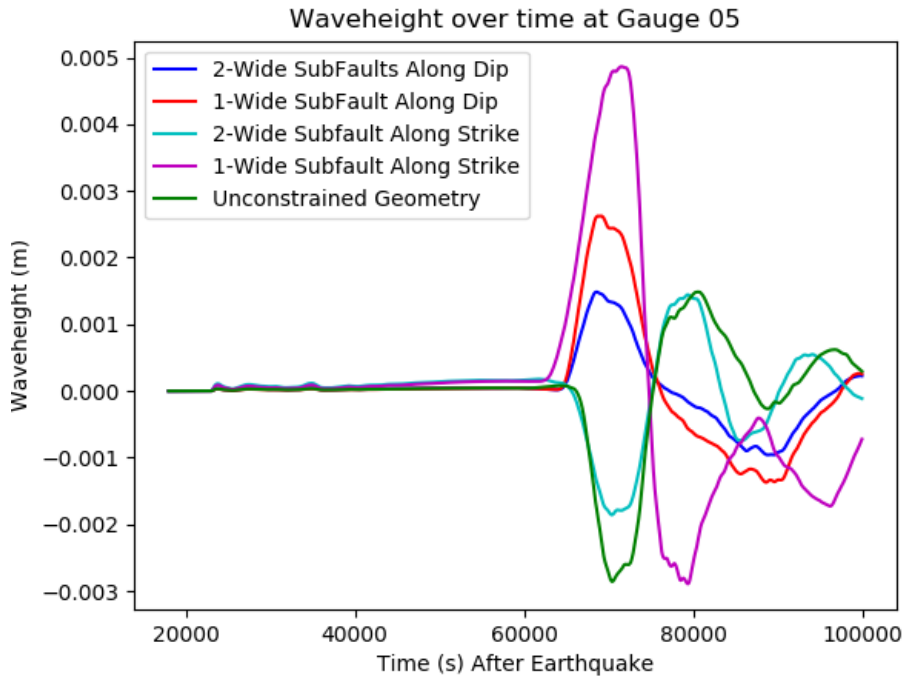


Figure 4.25: Wave height over time for the five different scenarios at Gauge 05 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.0044 m peak. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough was caused by the unconstrained geometry of sub-faults at -0.0028 m.

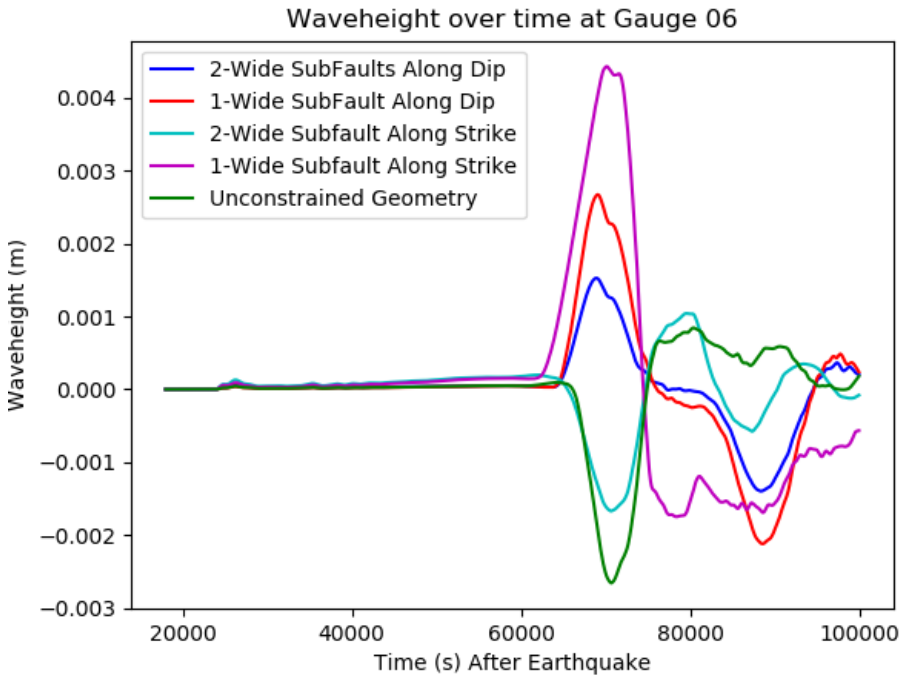


Figure 4.26: Wave height over time for the five different scenarios at Gauge 06 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.0045 m peak, and a -0.0027 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0027 m.

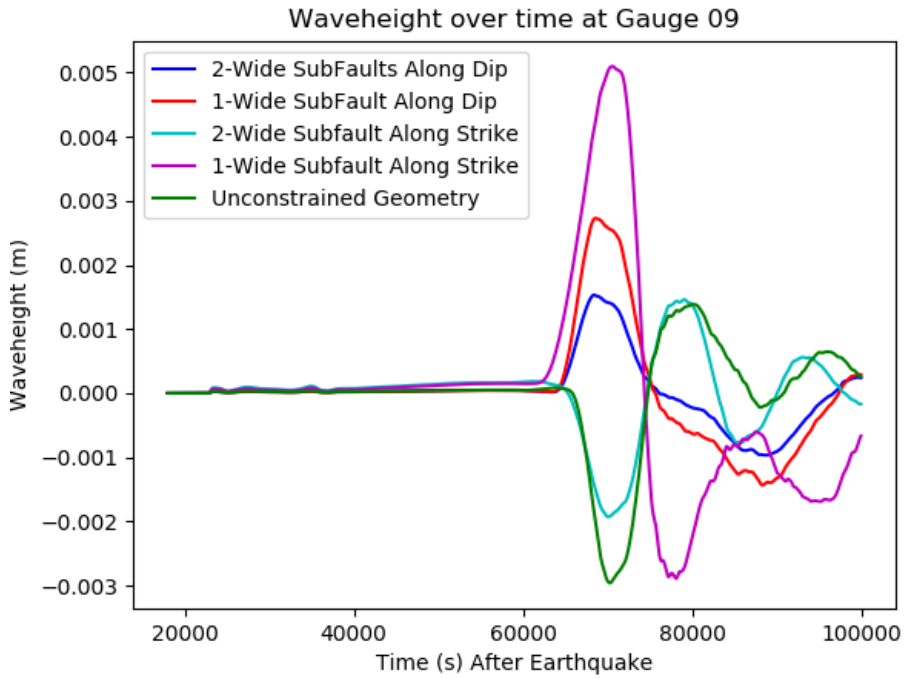


Figure 4.27: Wave height over time for the five different scenarios at Gauge 09 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak, and a -0.0028 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0029 m.

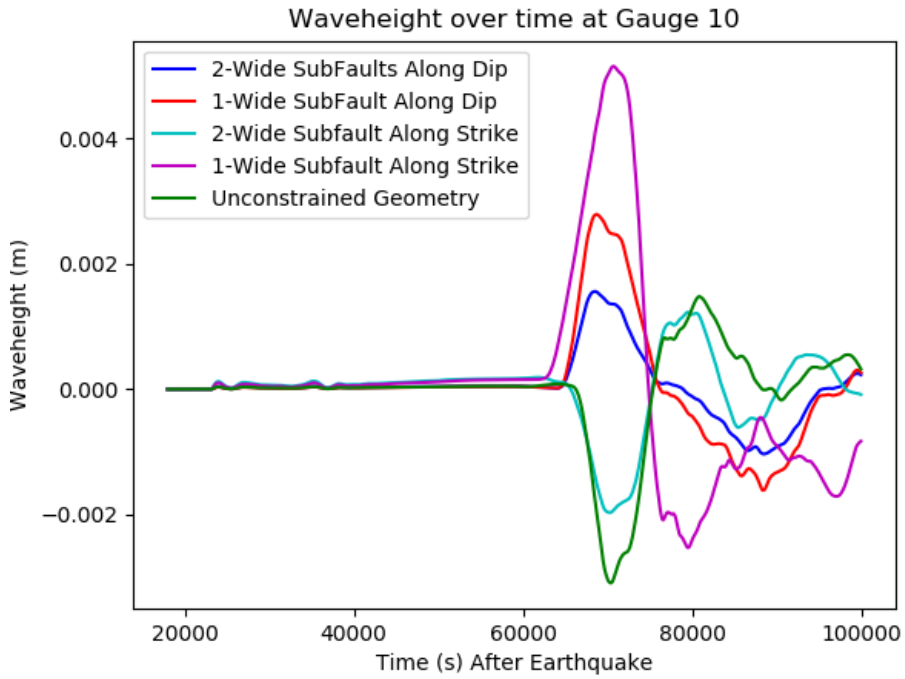


Figure 4.28: Wave height over time for the five different scenarios at Gauge 10 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak, and a -0.0027 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0028 m.

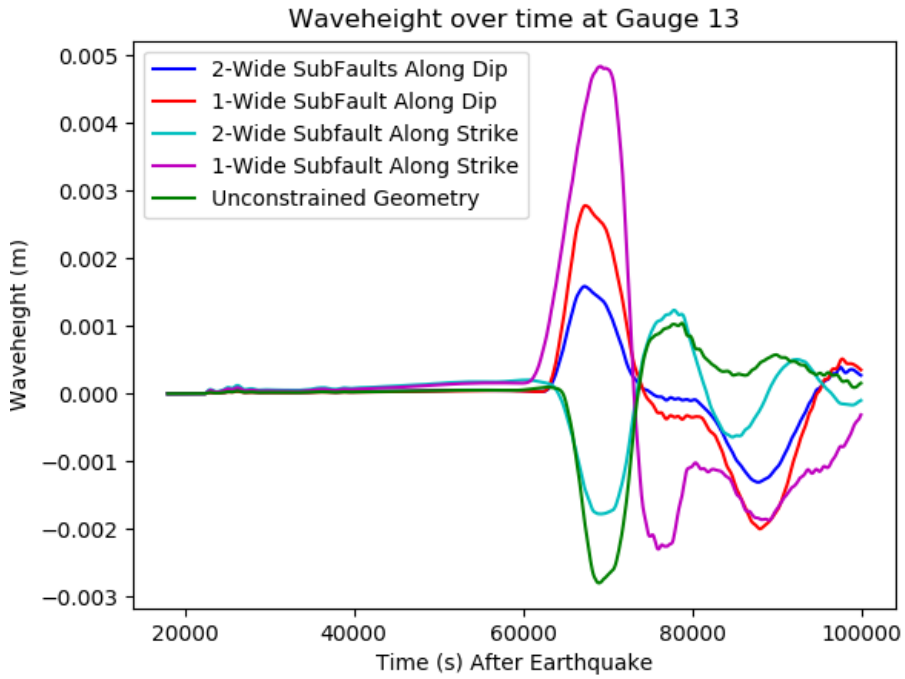


Figure 4.29: Wave height over time for the five different scenarios at Gauge 13 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak, and a -0.0026 m trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough, however, was caused by the unconstrained geometry of sub-faults at -0.0028 m.

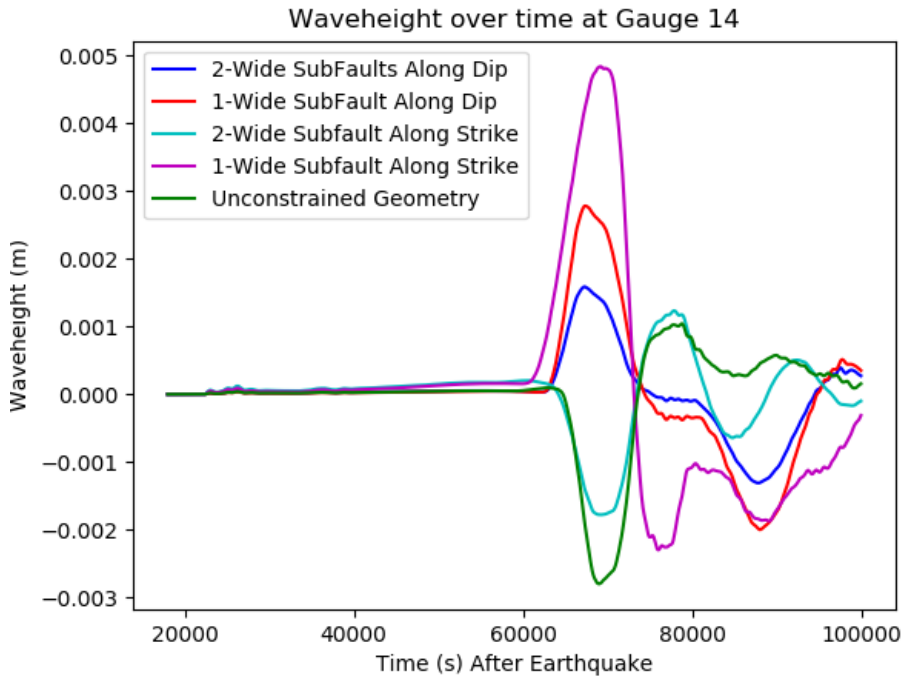


Figure 4.30: Wave height over time for the five different scenarios at Gauge 14 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The second trough, however, was caused by the unconstrained geometry of sub-faults at -0.0028 m.

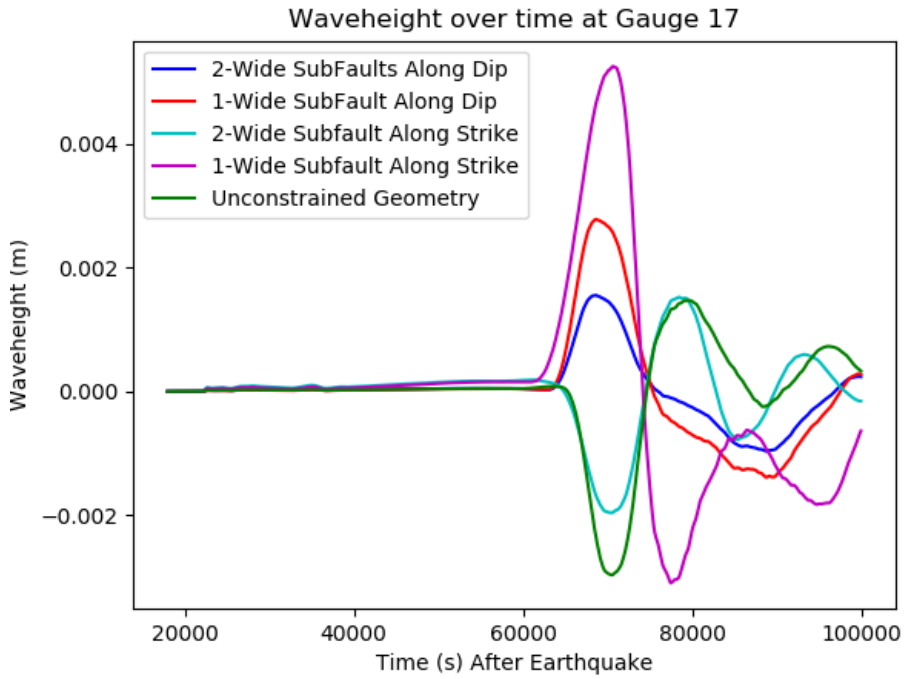


Figure 4.31: Wave height over time for the five different scenarios at Gauge 17 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough was also caused by the single wide row of sub-faults in an along strike geometry, at -0.0028 m.

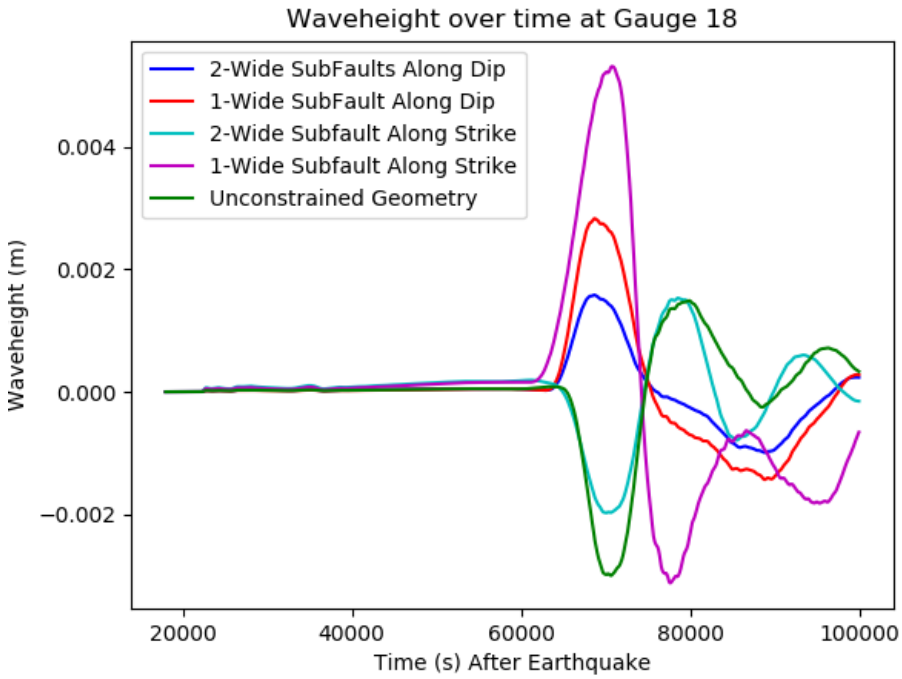


Figure 4.32: Wave height over time for the five different scenarios at Gauge 18 during the Chilean event. The largest peak was caused by a single wide row of sub-faults in an along strike geometry, with a minute 0.005 m peak, and a -0.003 trough. The second highest peak was caused by the single row of sub-faults in an along-dip geometry, with a height of 0.0027 m. The lowest trough was also caused by the single wide row of sub-faults in an along strike geometry, at -0.0028 m.

4.3 Tohoku-Oki 2011 Case Study

At 2:46 pm local time on March 11, 2011, a magnitude 9.0 to 9.1 earthquake occurred off the eastern coast of Japan, near the Tohoku peninsula, at 38.322 ^{o}N, 142.369 ^{o}E. It occurred at an estimated depth of 30 km , with estimated fault displacements of up to 40m along a 100 km rupture (Ammon, Lay, Kanamori, & Cleveland, 2011). The earthquake and resulting tsunami caused a combined 19,000 deaths and missing persons as well as $211 billion USD in damages to over 804 thousand residential buildings, causing the displacement of 342 thousand people nearly a full year after the event (Kajitani, Chang, & Tatano, 2013). Japan is located near the Kurile, Izu-Bonin, and Japan trenches described by NOAA's SIFT database, where the Pacific plate is subducting under the Okhotsk plate, a micro plate that was formerly part of the North American continental plate (Seno, Sakurai, & Stein, 1996), at a rate between 7.9 and 9.2 cm/yr (Sella, Dixon, & Mao, 2002). This area, being in a subduction zone, is prone to large earthquakes, as shown by 7.4 magnitude earthquake on December 21, 2010, the 8.1 magnitude on January 13, 2007 in the Kuril Islands, and again in the Kuril Islands with an 8.3. The Tohoku-Oki earthquake is one of the strongest earthquakes in recent history, and is used as a case study in tsunami generation, as it is capable of generating tsunamis devastating to both life and economics.

4.3.1 Tohoku-Oki Modelling Results

The automatic fault geometry constraining generated the five variable geometries as described earlier in Section 3.2.1, the parameters of which can each be seen in Tables A.6 through A.10. As previously described in Section 4.2, the sea floor is deformed, as described by an Okada model, displacing the water above. A model time of 0.08 hours is

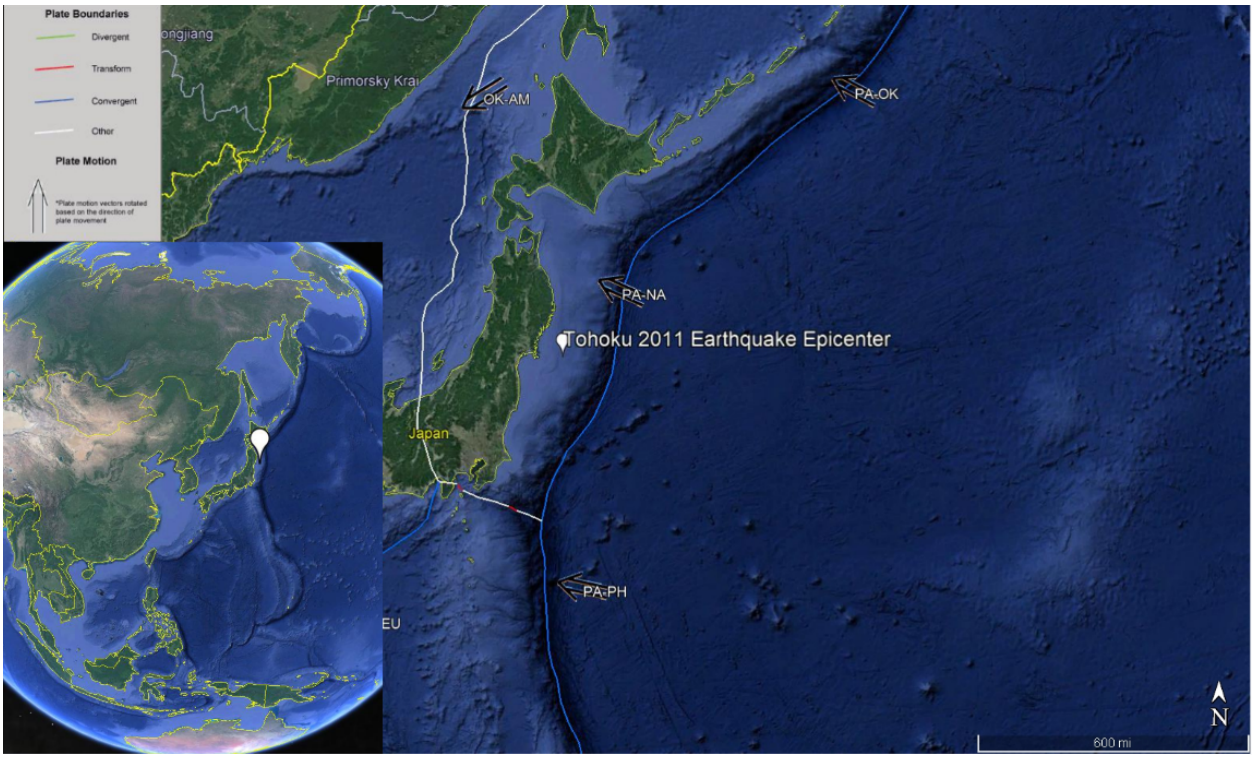
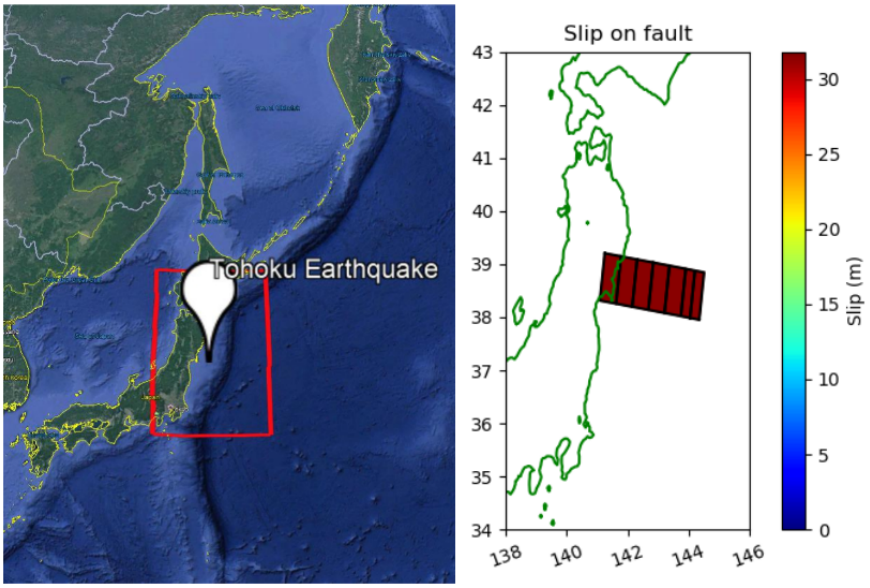


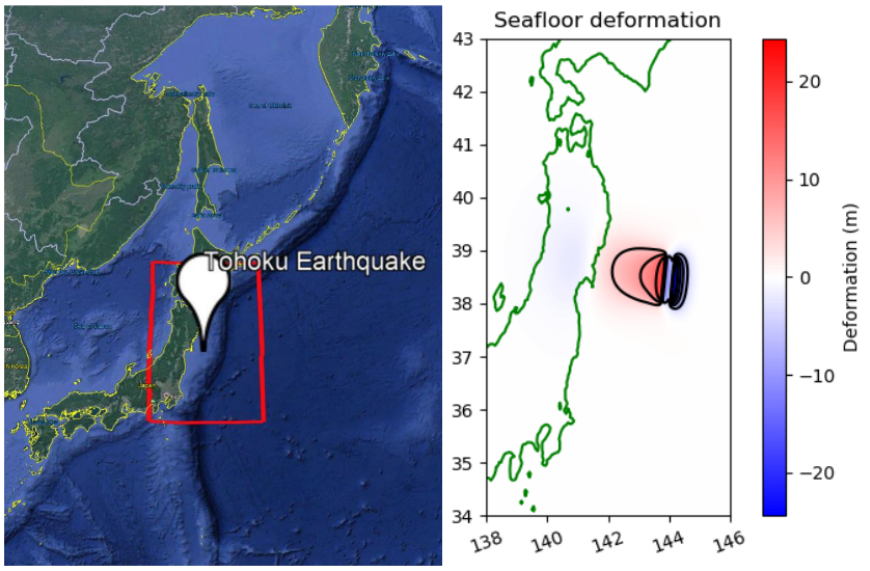
Figure 4.33: The epicenter of the Tohoku 2011 earthquake from a Google Earth .KMZ file, including an overlay from the USGS showing tectonic plates and their interactions, where the Pacific plate can be seen subducting under the North American plate. The Pacific plate can also be seen subducting under the Phillipine plate in the south, and the Okhotsk plate in the north. This is the location input of the geometry constraining python script, with a location of 38.322 ^{o}N, 142.369 ^{o}E, located in the Japan Subduction Zone, with the Kuril Subduction Zone to the north and the Izu-Bonin Subduction Zone to the south.

used again, as it was in the Chilean example. In this failure geometry, it appears that there is primarily a raising of the water surface elevation with little to no depreciation in the normal water surface. Conversely, Figure 4.14, a failure geometry of two along-dip sub-faults, which is represented by the Okada models in various figures below, as well as their corresponding fault geometries, showing depreciation of the water elevation changes. Figures 4.40a and 4.40b are geometries that are constrained to the along strike direction in a single and double row, respectively, but both expand to the Kuril Subduction Zone to the north. Each contains alternating areas where the water column raises and lowers,

caused by the deformation of the Okada models seen in Figures 4.36b and 4.37b. Lastly, Figure 4.41a is the unconstrained geometry, where once again the nearest number of required sub-faults are located. This figure shows that there is a significant displacement in both the positive and negative directions of the water column, caused by the displacement of the sea floor as seen in the Okada model Figure 4.38b. Each model has a resolution of 1 arc-minute at the coarsest, and can be refined for higher resolutions, as dictated by the user. In the case of Figures 4.39 through 4.41, the resolution was forced to maximum in GeoClaw around the study area for a clearer figure of the water column behavior soon after the earthquake rupture.

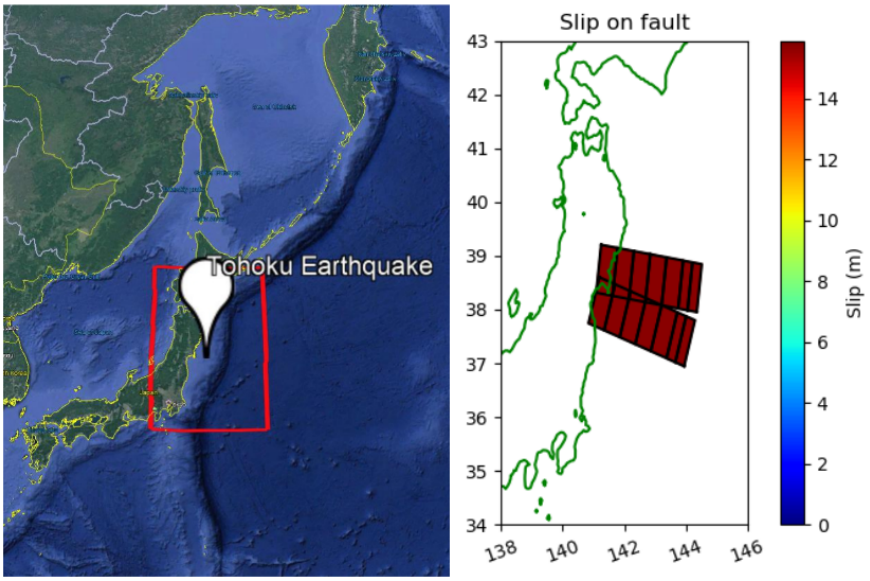


(a)

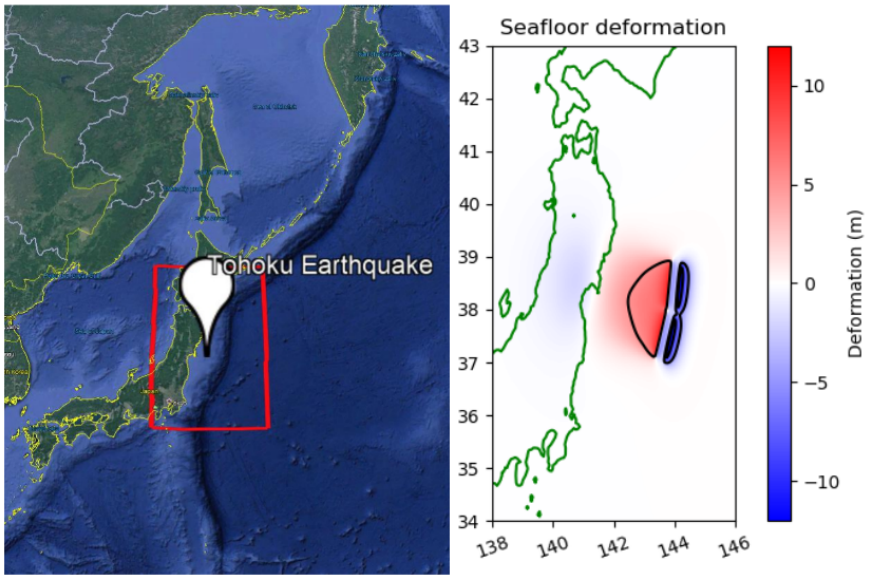


(b)

Figure 4.34: The distribution of sub-faults and deformation from an Okada model of a single row of along-dip geometry, with the model having a 5 m contour interval. Sub-figure a shows the distribution of sub-faults used in this geometry, while b shows the results of the corresponding okada model. The sea floor deformation shown here has a maximum elevation change of 20 m in uplift, and 23 m in subsidence.

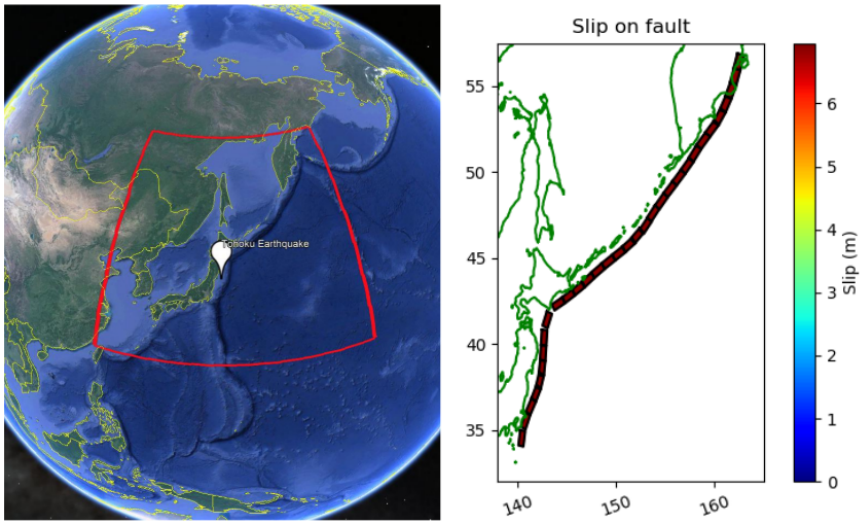


(a)

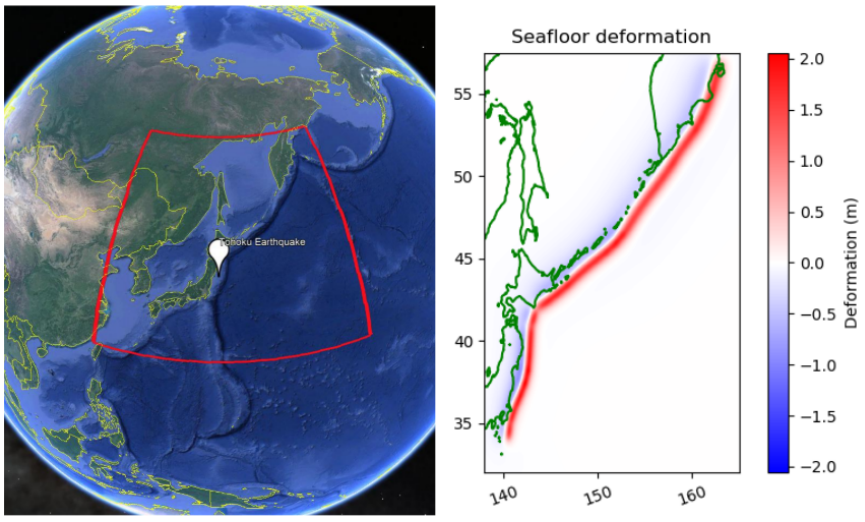


(b)

Figure 4.35: The distribution of sub-faults and deformation from an Okada model with two rows of along-dip geometry, with the model having a 5 m contour interval. Sub-figure a shows the sub-fault distribution, while sub-figure b shows the resulting deformation from the corresponding Okada model. The sea floor deformation has a maximum elevation change of 9 m in uplift, but up to 12 m of subsidence.

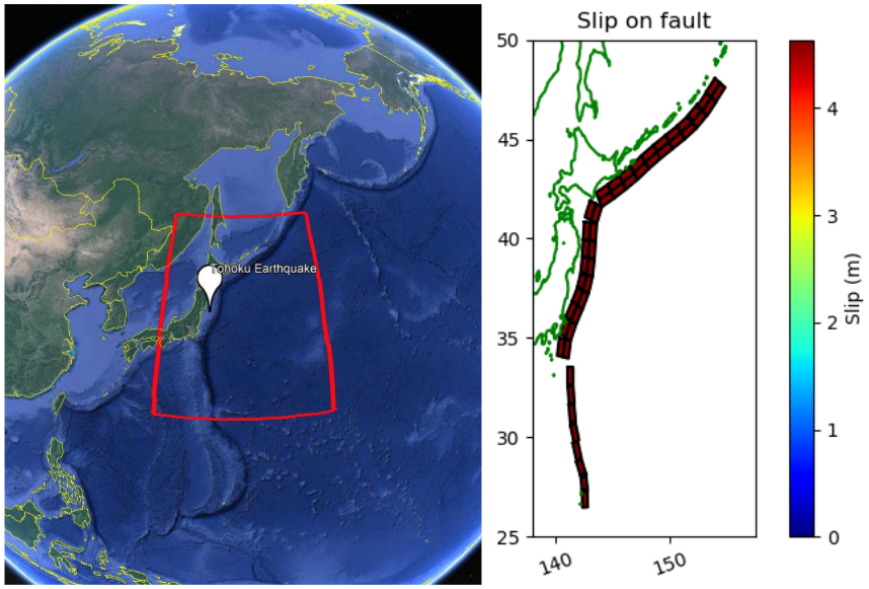


(a)

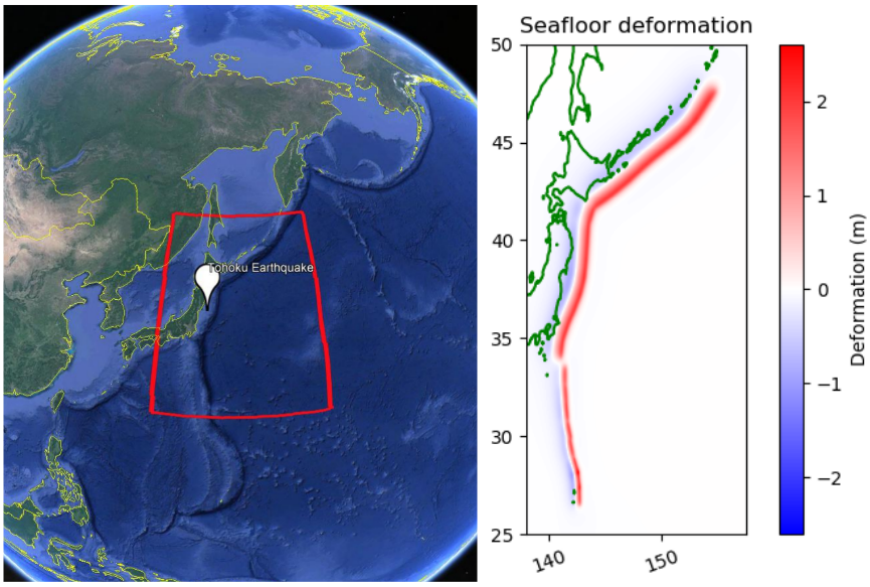


(b)

Figure 4.36: Sub-figure a shows the distribution of sub-faults determined in an along-strike, single wide geometry, with sub-figure b showing the sea floor deformation. The deformation ranged from 2m of uplift to 1m of subsidence.

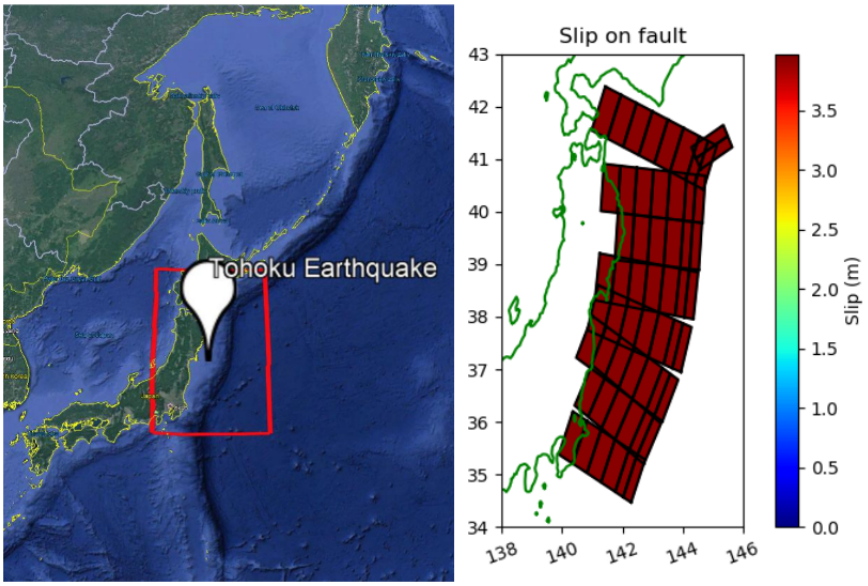


(a)

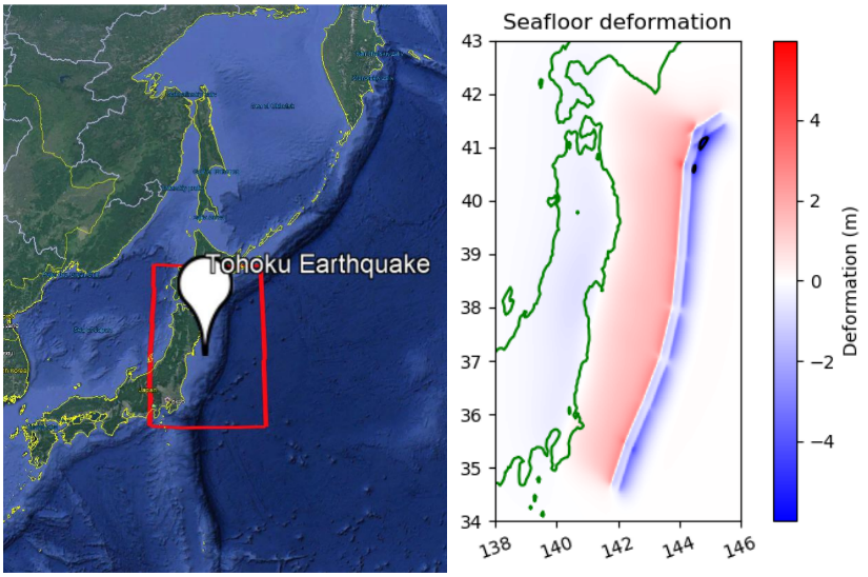


(b)

Figure 4.37: Sub-figure a shows the distribution of sub-faults determined in an along-strike, two-wide geometry, with sub-figure b showing the sea floor deformation. The deformation ranged from 1.5 m of uplift to 0.5 m of subsidence.

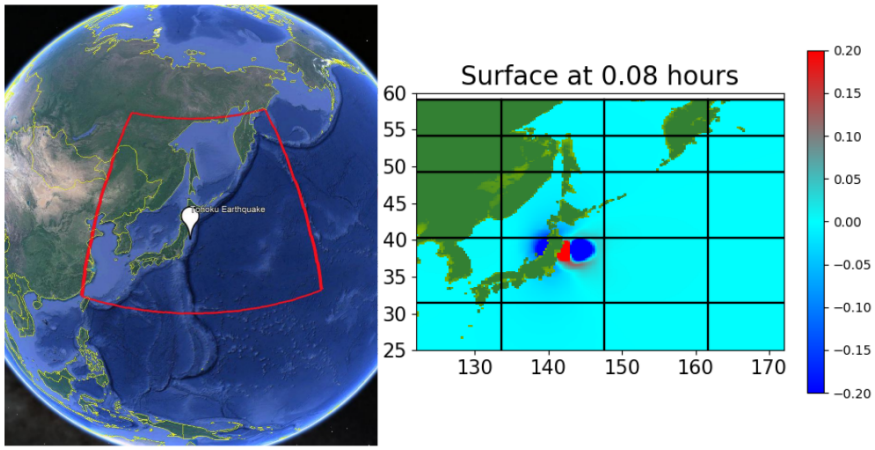


(a)

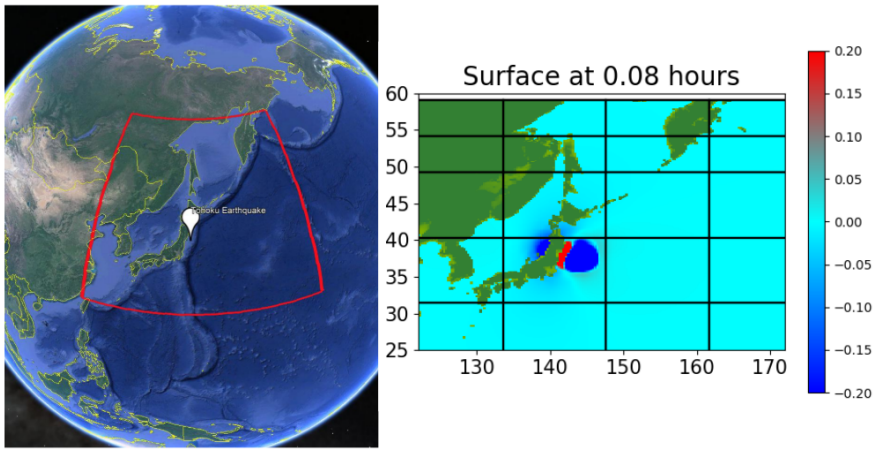


(b)

Figure 4.38: The distribution of sub-faults and deformation from an Okada model with a geometry determined by the Haversine Equation. Sub-figure a shows the distribution of the resulting sub-faults, while sub-figure b shows the resulting Okada model of the sub-faults, with a maximum uplift of 3 m and maximum subsidence of 5 m.

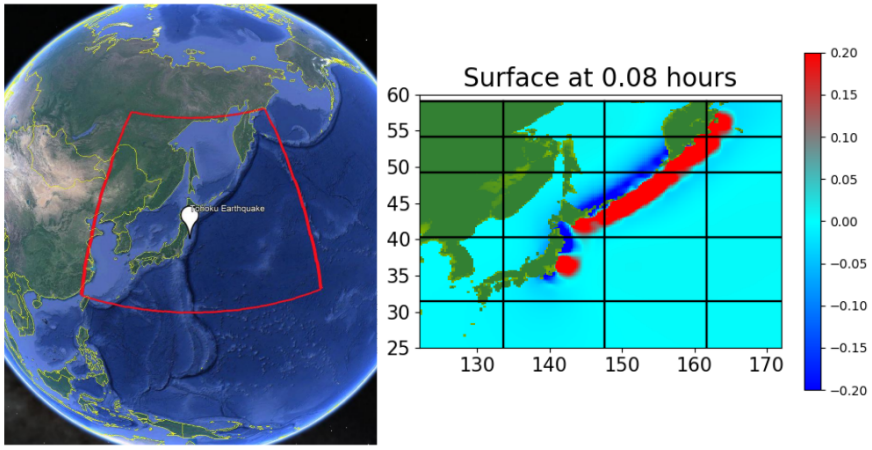


(a)

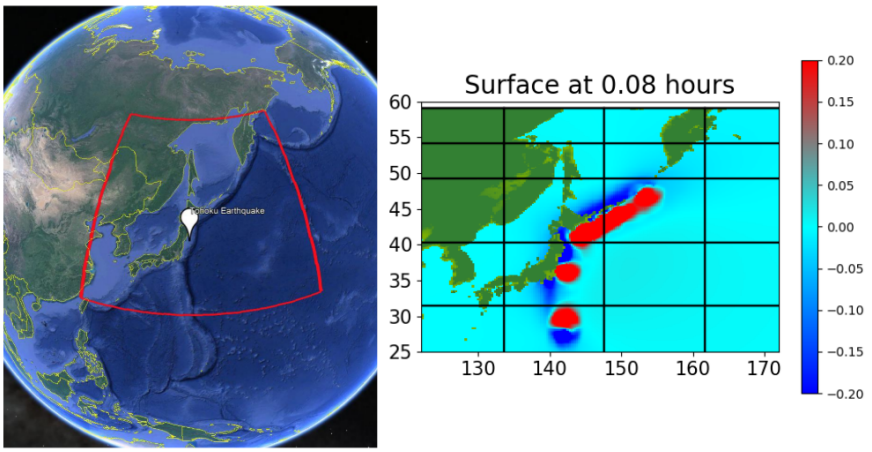


(b)

Figure 4.39: The two along-dip geometries, with sub-figure a showing the single row of sub-faults, and sub-figure b showing two rows of along-dip geometry. The two rows of along-dip geometry shows a large volume of water undergoing sea level lowering, whereas the single row does not show any drop in water level.

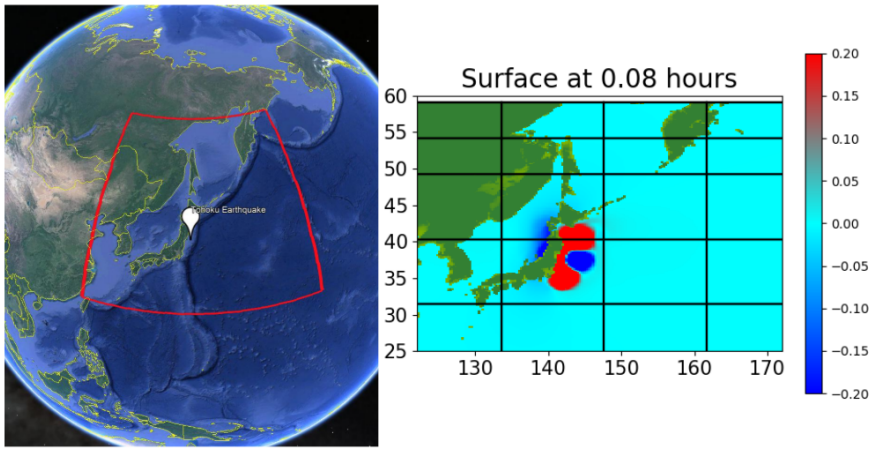


(a)



(b)

Figure 4.40: The two along-strike geometries of the Chilean 2010 earthquake. Sub-figure a shows both water level raising and lowering along the majority of the western coast of South America, where as sub-figure b shows a more concentrated area of water level change that is not as extensive.



(a)

Figure 4.41: The water elevation change due to the unconstrained geometry determined by the Haversine Equation. It shows concentrated areas of water level lowering both in the north and south, with the central area showing a large amounts of water level raising

4.3.2 Wave Gauges for the Tohoku-Oki 2011 Tsunami

As stated in Section 4.1, the wave heights for each model were recorded at 18 wave gauge locations, recording the mass flux in the x, y, and z directions as well as wave height. The wave heights are displayed below, in Figures 4.21 through 4.32. Some gauges, such as 04, 07, 08, 11, 12, 15, and 16, showed a consistent elevation, caused by the adaptive mesh refinement and resolution causing the locations that these wave gauges are located at to be on land. This could be overcome with moving the locations, but these particular locations were selected as they were the same wave gauges that are implemented in Arcos & LeVeque (2015). Figure 4.42, seen below, is Gauge 01 from Arcos & LeVeque (2015), and can be used to compare to Figure 4.43, with the red line showing the results of a GeoClaw model, and the blue line showing the recorded tsunami wave height over time.

As can be seen in the following figures, each distinct geometry had a significantly different wave pattern. In general, the largest waves were caused by the two sub-fault wide, along-dip geometry, with an average maximum wave height of .216 m. Fairly consistently,

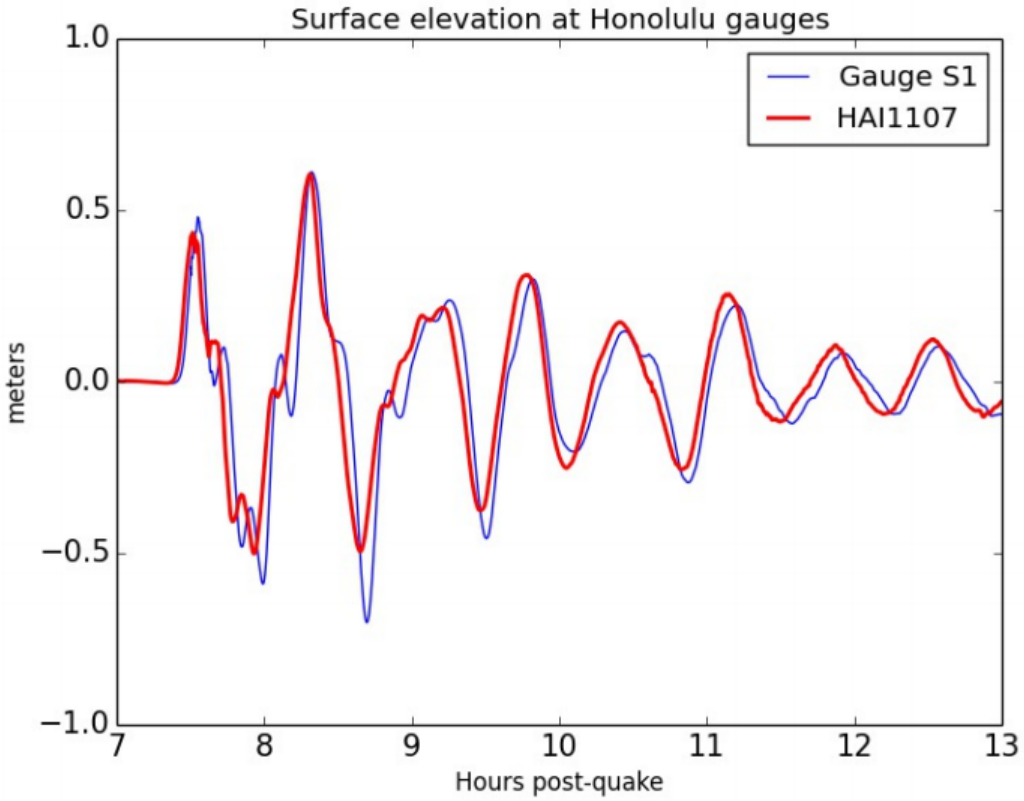


Figure 4.42: The wave height over time for Gauge 01, as modeled by Arcos (2015). When compared to Figure 4.43, we can see that the best comparison for models seems to be the single row of along-strike sub-faults, although they seem to be off by a factor of 10. A 2 wide row of along-dip geometry has a larger wave amplitude, although it is still a factor of 3 smaller than the measured data from Arcos (2015). This figure also takes into consideration the tidal affects of the area, however, and may cause more drastic wave height changes.

the least impactful geometry was the unconstrained sub-faults, with consistently nominal wave heights. All five sub-fault geometries, despite having the same magnitude, have wildly different wave height maximums, minimums, and overall propagation parameters. This is unsurprising, as some of these events are highly unlikely to occur, such as the single row of sub-faults in an along dip geometry. Cheung (2013), in which also models the Tohoku event around Hawaii, can be used to compare wave heights to the models

(Cheung, Bai, & Yamazaki, 2013).

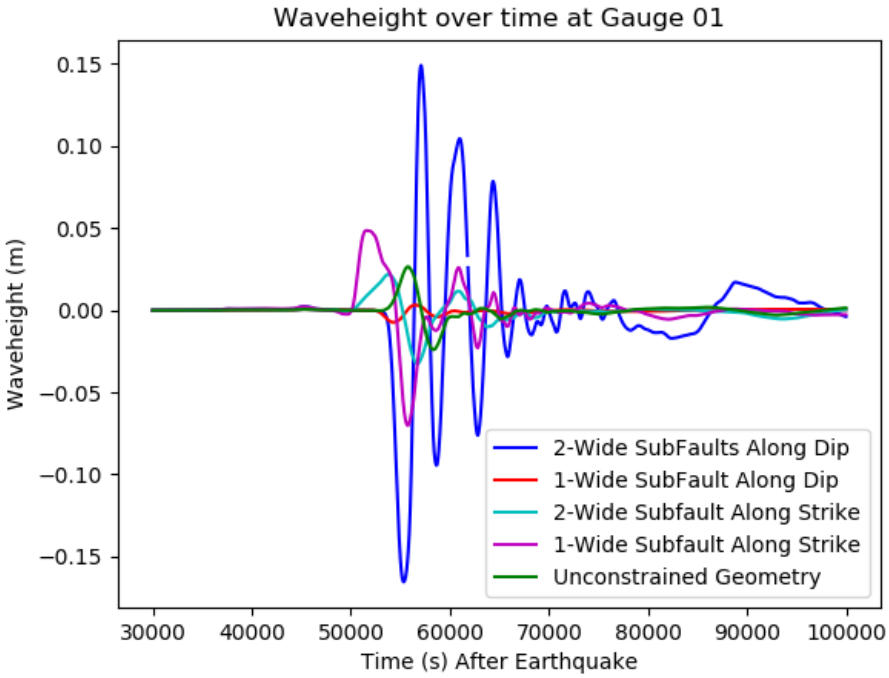


Figure 4.43: The wave height over time for the five different scenarios at Gauge 01 during the Tohoku event. The largest wave showed an increase of 0.15 m and a largest decrease of -0.16 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-dip geometry at 0.05m at the wave peak and -0.07 m at the trough. The single row of along-strike sub-faults caused the least amount of difference from zero.

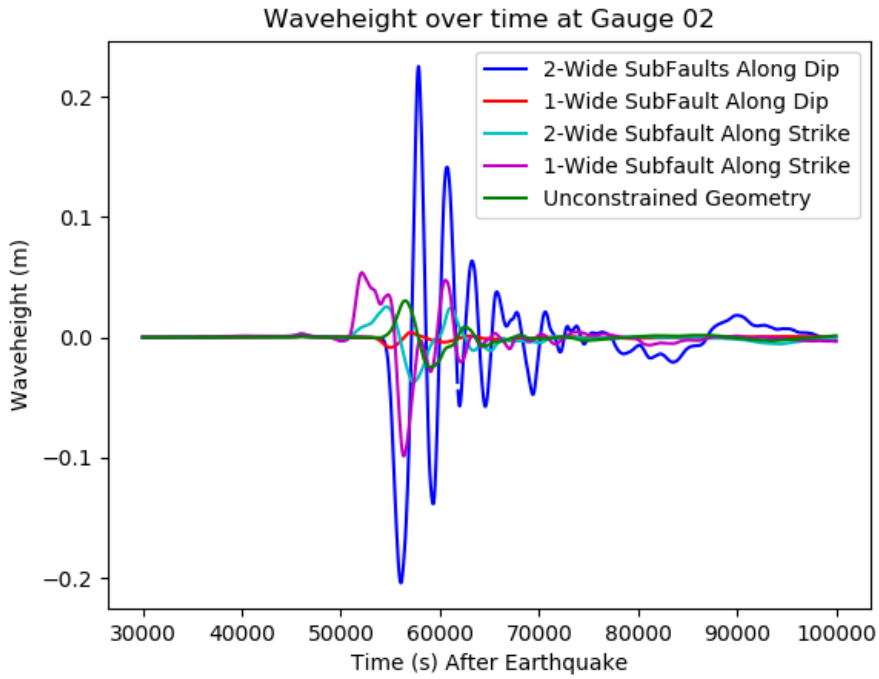


Figure 4.44: The wave height over time for the five different scenarios at Gauge 02 during the Tohoku event. The largest wave showed an increase of 0.23 m and a largest decrease of -0.21 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.05 m at the wave peak and -0.1 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

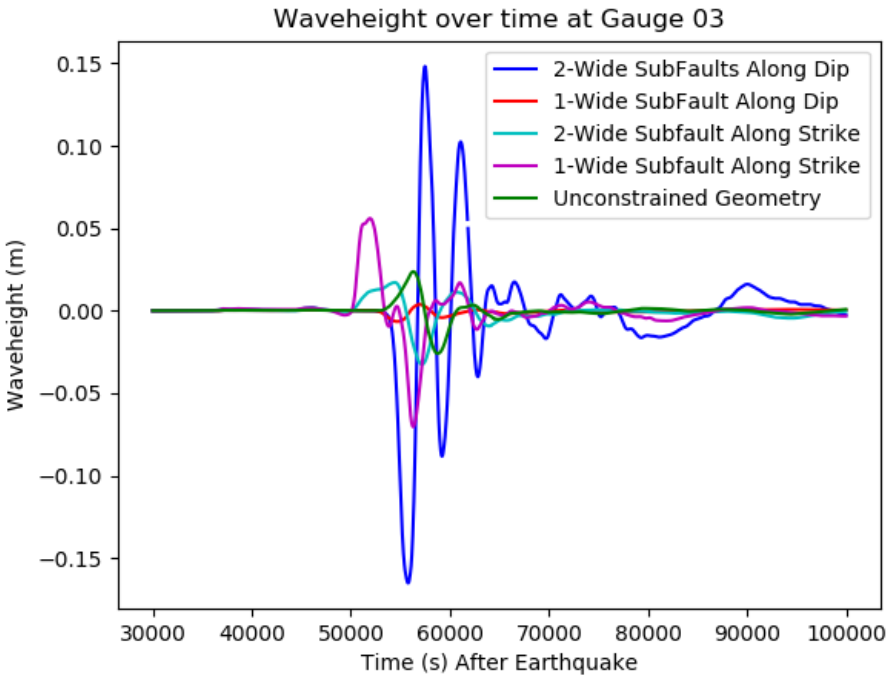


Figure 4.45: The wave height over time for the five different scenarios at Gauge 03 during the Tohoku event. The largest wave showed an increase of 0.15 m and a largest decrease of -0.16 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.06 m at the wave peak and -0.07 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

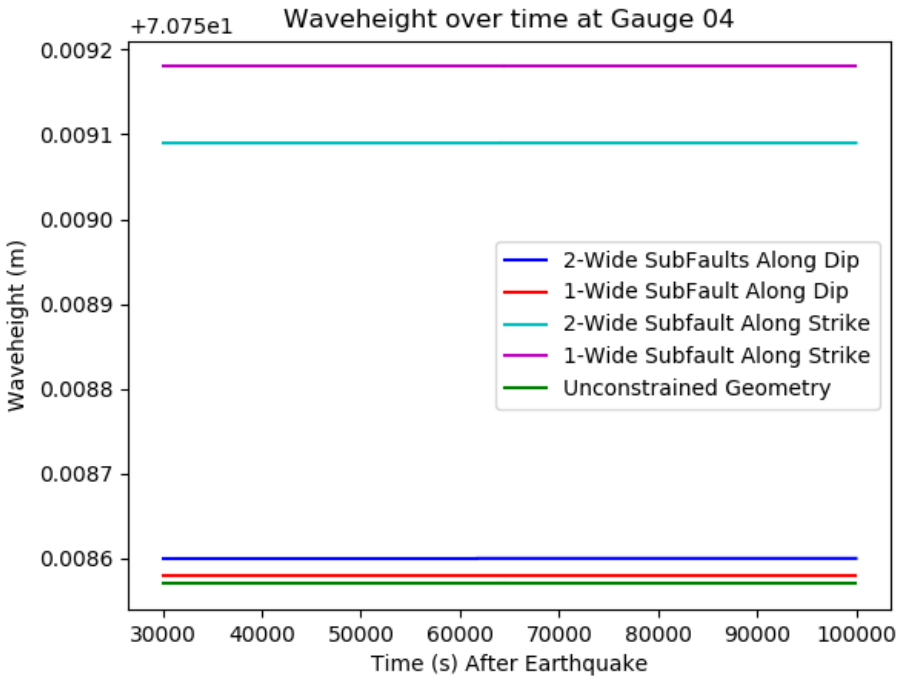


Figure 4.46: The wave height over time for the five different scenarios at Gauge 04 during the Tohoku event. The adaptive mesh refinement and grid system used in GeoClaw calculated the cell this gauge was in as land. Thus, each geometry shows a constant elevation, and in this case, an elevation of just over 70 m. It is unclear why each geometry had a different constant elevation, although variations are in the mm range.

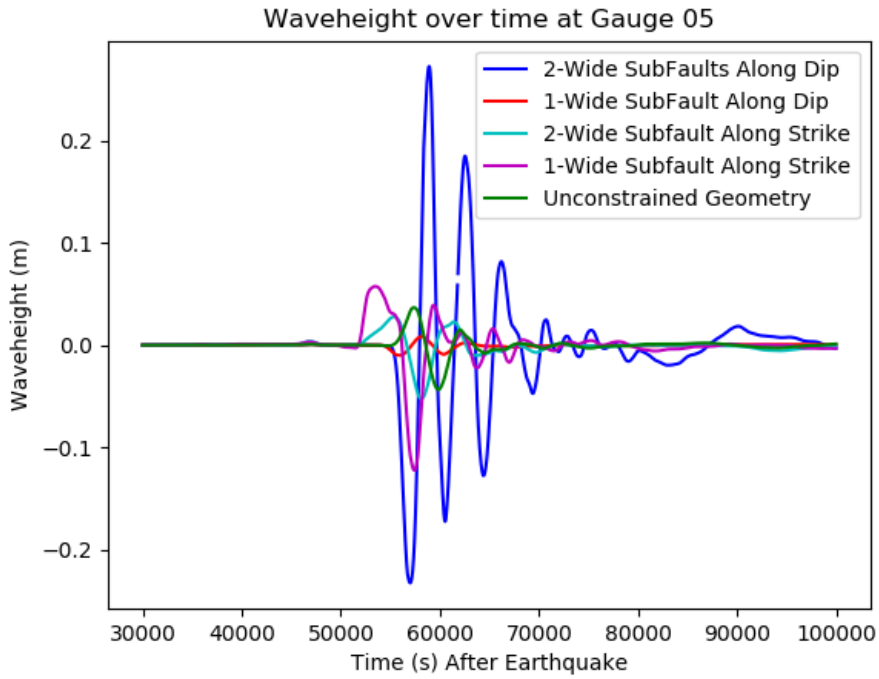


Figure 4.47: The wave height over time for the five different scenarios at Gauge 05 during the Tohoku event. The largest wave showed an increase of 0.27 m and a largest decrease of -0.23 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.06 m at the wave peak and -0.12 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

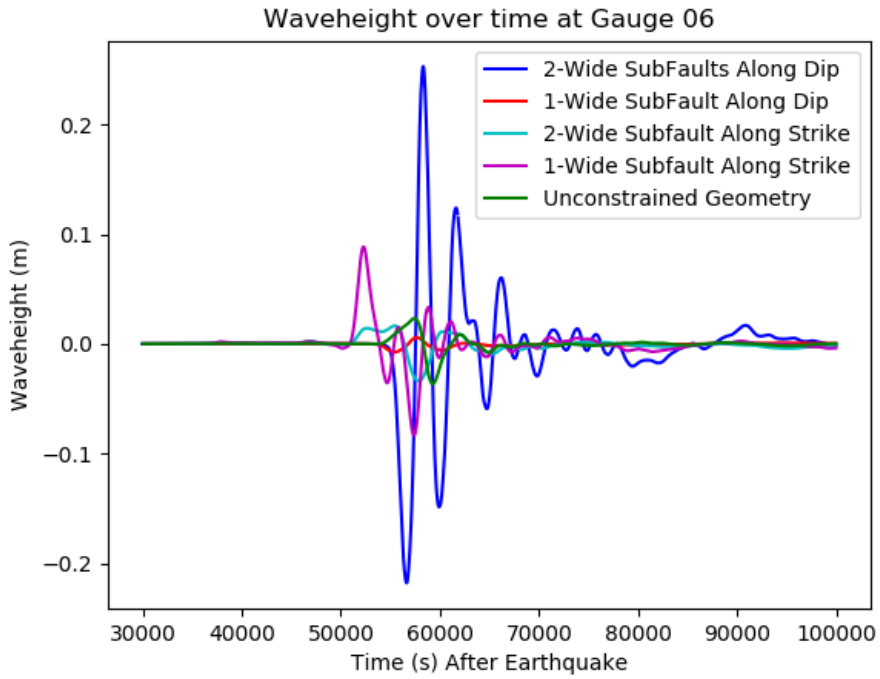


Figure 4.48: The wave height over time for the five different scenarios at Gauge 06 during the Tohoku event. The largest wave showed an increase of 0.27 m and a largest decrease of -0.21 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.1 m at the wave peak and trough at -0.08 m. The unconstrained sub-faults caused the least amount of difference from zero.

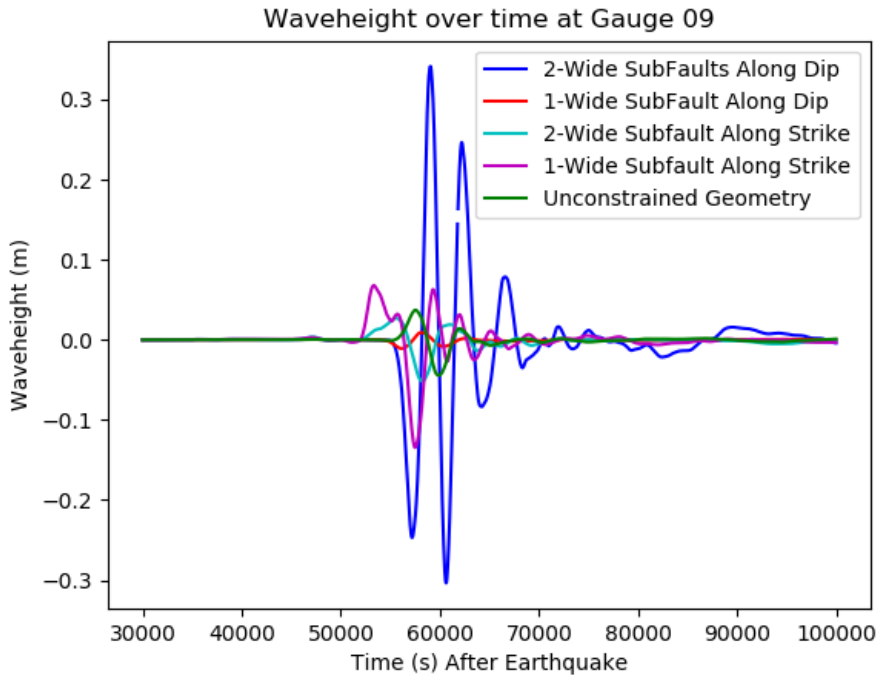


Figure 4.49: The wave height over time for the five different scenarios at Gauge 09 during the Tohoku event. The largest wave showed an increase of 0.35 m and a largest decrease of -0.32 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.9 m at the wave peak and -0.15 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

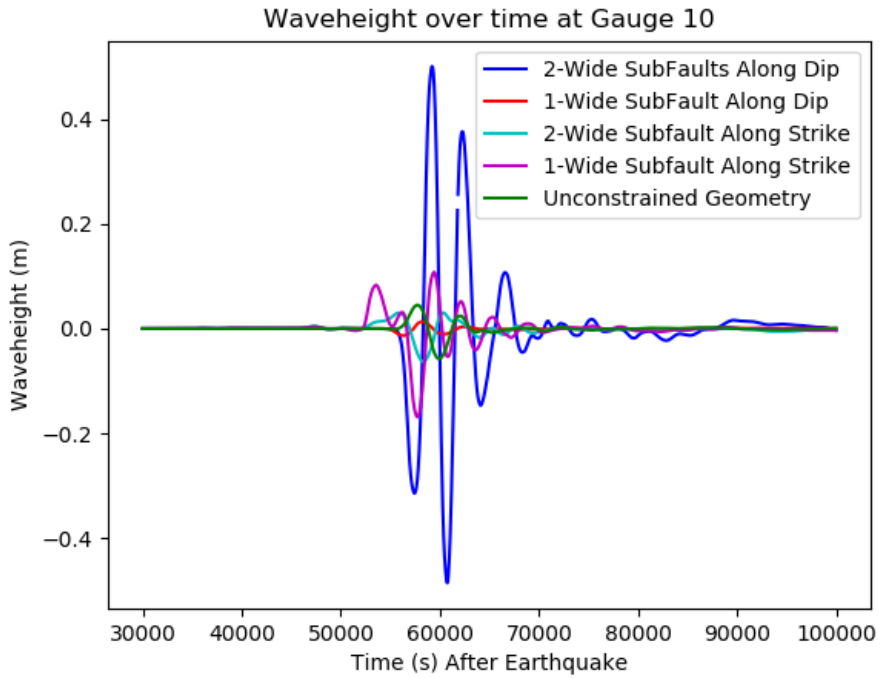


Figure 4.50: The wave height over time for the five different scenarios at Gauge 10 during the Tohoku event. The largest wave showed an increase of 0.48 m and a largest decrease of -0.46 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.9 m at the wave peak and -0.16 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

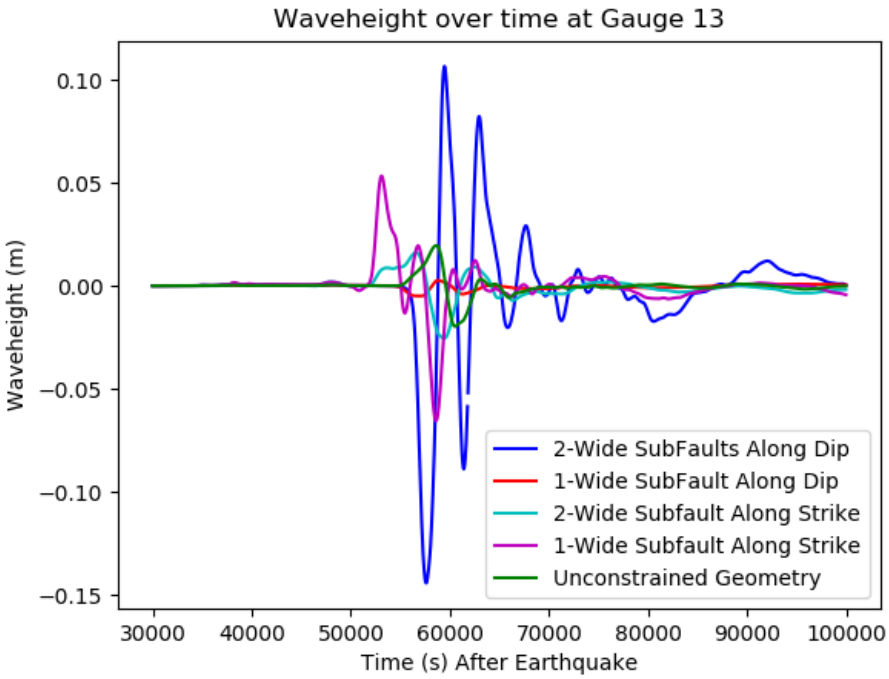


Figure 4.51: The wave height over time for the five different scenarios at Gauge 13 during the Tohoku event. The largest wave showed an increase of 0.11 m and a largest decrease of -0.14 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.06 m at the wave peak and -0.12 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

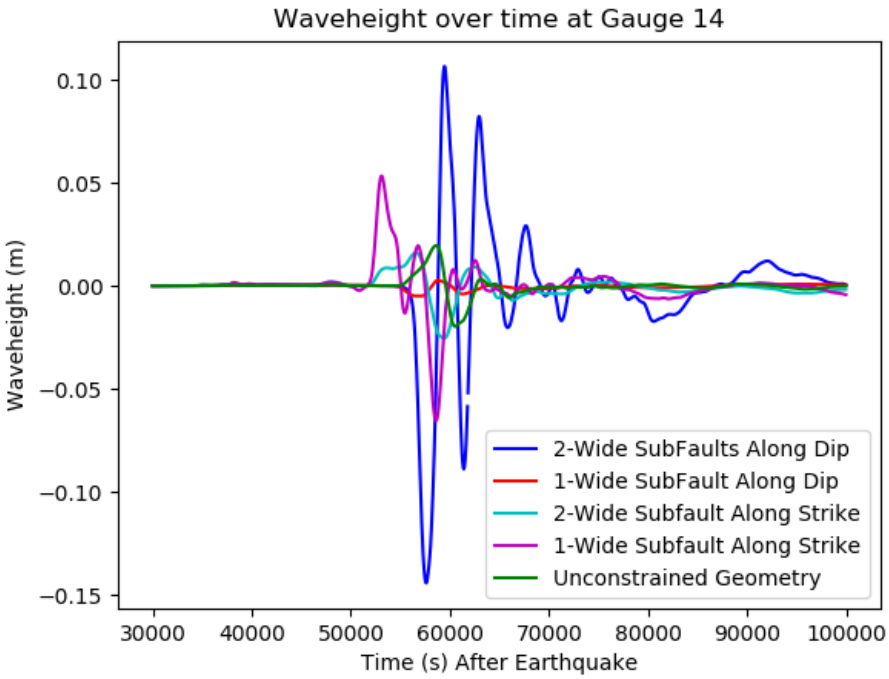


Figure 4.52: The wave height over time for the five different scenarios at Gauge 14 during the Tohoku event. The largest wave showed an increase of 0.10 m and a largest decrease of -0.14 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.05 m at the wave peak and -0.07 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

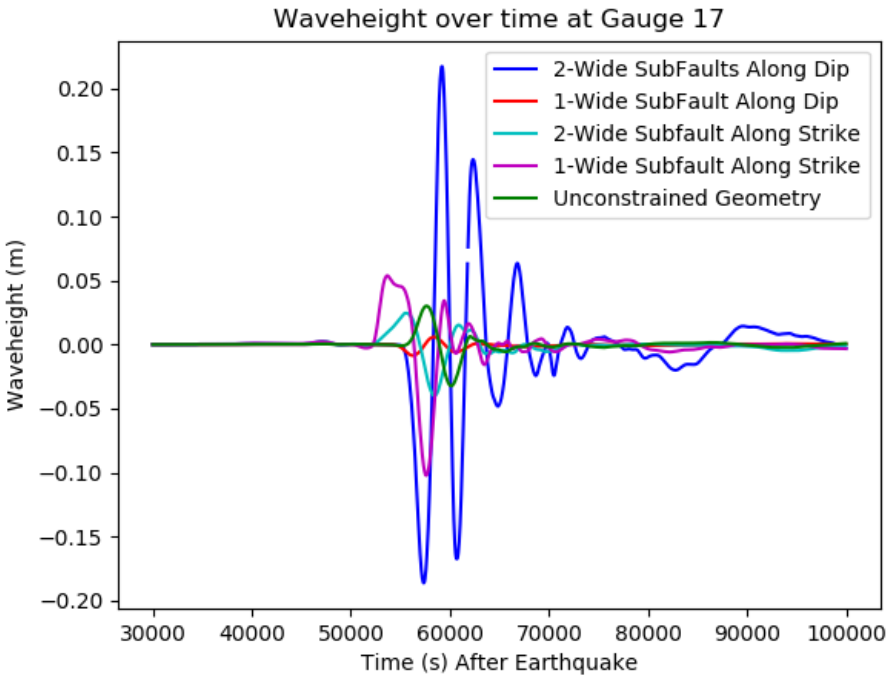


Figure 4.53: The wave height over time for the five different scenarios at Gauge 17 during the Tohoku event. The largest wave showed an increase of 0.23 m and a largest decrease of -0.18 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.06 m at the wave peak and -0.12 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

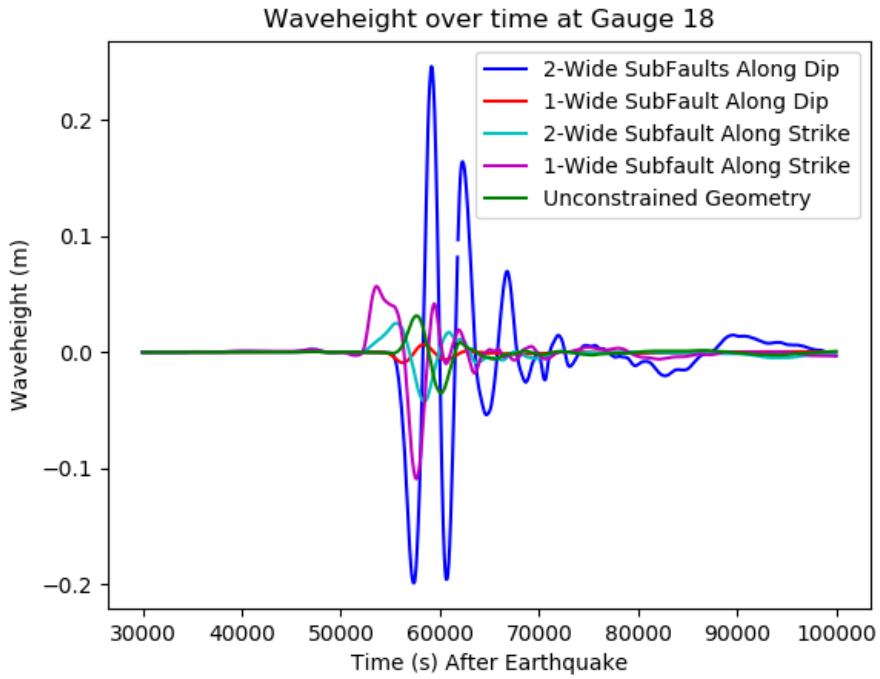


Figure 4.54: The wave height over time for the five different scenarios at Gauge 18 during the Tohoku event. The largest wave showed an increase of 0.25 m and a largest decrease of -0.20 m, caused by the two rows of sub-faults in the along-dip geometry. The next largest wave height was caused by the single wide row of sub-faults in the along-strike geometry at 0.05 m at the wave peak and -0.11 m at the trough. The single row of along-dip sub-faults caused the least amount of difference from zero.

Chapter 5

Conclusions

The purpose of this study was to create a series of programs to model earthquake generated tsunamis for use in the now-casting to help aid scientists in the determination of the most probable, generalized rupture model of a given earthquake. A system in which sub-faults are implemented has been previously conducted, as evidenced by Gica et al., 2008, and the creation of the NOAA SIFT database, and several studies, as discussed in Section 2.2, show that tsunamis are highly sensitive to the geometry of fault failure. Using GeoClaw, a series of tsunami models are successfully created, using various fault failure geometries estimated in a separate Python code. These geometries, described in Sections 4.2.1 and 4.3.1, can be seen to have a significant difference in propagation behavior immediately after the earthquake. As seen in Sections 4.3.2 and 4.2.2, these variations in fault geometry also have a significant impact on the wave propagation behavior, which in turn, as seen by Figures 4.43 through 4.54 for the Tohoku tsunami and 4.21 through 4.32 for the Chilean tsunami, has an impact on wave height. Overall, this study has been successful, with the significant exception of the magnitude of wave height of the Chilean case study. There are other limitations, which are addressed in Section 6. These include a

variable slip, an inability of the program to change subduction zone names provided by NOAA, and at at times, unreasonable amounts of concentrated slip, as seen in the Tohoku case study's geometry of a single wide, along-dip sub-faults.

The lack of correlation in these models to observed tsunami behavior is not necessarily indicative of this method's abilities to estimate an earthquake's source, but is indicative of the sub-fault geometries previously modeled abilities to replicate the Tohoku event. Work from Arcos & LeVeque (2015) show GeoClaw's ability to model the event, and thus this method is very likely viable, and that the poor correlation seen above is likely due to the limited geometries chosen for earthquake fault failure.

Despite the limitations, the code created during this study provides a good starting point to develop a more robust model. The models take minutes to hours, depending on the resolution and number of time steps a user determines, rather than days to weeks for seismic waveform inversion to be resolved. Although the programs created and edited to be implemented in these models of GeoClaw are not meant to be a replacement for using waveform inversion or other, more accurate means of fault failure determination, these programs will allow for the faster modelling of an event at a lower accuracy, and is intended to supplement rather than replace more thorough models.

Chapter 6

Future Work

There are a series of problems to which solutions could be implemented in future work on this project. A variable slip distribution is a necessity, along with increased number of geometry variations of the sub-faults, as this would more accurately represent the behavior of fault failure, and could greatly impact the propagation and generation of tsunami waves, especially in far-field locations. A 'best guess' would have to be implemented, however, as the distribution of slip in an earthquake rupture is unavailable for some time following the event, and these scripts are intended to be implemented in a 'nowcasting' style, where the event has recently occurred within the past few minutes and hours. Another shortcoming of the current implementation of the automation of the assignment of the geometry is that it is only capable of constraining geometry within the same subduction zone. As seen in the case study of the Tohoku-Oki earthquake, and displayed in Figure **??**, it would be more reasonable to assume that the fault failure geometry would be centered around the earthquake epicenter, rather than be offset to the north to stay in the same 'subduction zone' that is dictated by NOAA's SIFT sub-fault database. This is caused by the manipulation of sub fault names, and the names of sub faults vary

significantly from those of other subduction zones, thus making this method difficult to implement across different subduction zones. There is another shortcoming in the form of excessive slip in the along-dip geometries in larger earthquakes. The Tohoku-Oki earthquake contains slip for up to nearly 40 m in the entire rupture. While this amount of slip is viable for smaller sections of sub-faults, it is unreasonable that an entire rupture zone would consist of such a high slip distance.

References

Ammon, C. J., Lay, T., Kanamori, H., & Cleveland, M. (2011). A rupture model of the 2011 off the pacific coast of tohoku earthquake. *Earth, Planets and Space*, *63*(7), 33.

Annunziato, A., Ulutas, E., & Titov, V. V. (2009). Tsunami model study using jrc-swan and noaa-sift forecast methods. In *International symposium on historical earthquakes and conservation of monuments and sites in the eastern mediterranean region 500th anniversary year of the 1509, book of proceedings, istanbul* (pp. 131–141).

Arcos, M., & LeVeque, R. J. (2015). 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.

Barrientos, S. E., & Ward, S. N. (1990). The 1960 chile earthquake: inversion for slip distribution from surface deformation. *Geophysical Journal International*, *103*(3), 589–598.

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

Cheung, K. F., Bai, Y., & Yamazaki, Y. (2013). Surges around the hawaiian islands from the 2011 tohoku tsunami. *Journal of Geophysical Research: Oceans*, *118*(10),

5703–5719.

Duputel, Z., Rivera, L., Kanamori, H., & Hayes, G. (2012). W phase source inversion for moderate to large earthquakes (1990–2010). *Geophysical Journal International*, *189*(2), 1125–1147.

George, D. (2011). 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.

George, D. L., & Iverson, R. M. (2011). A two-phase debris-flow model that includes coupled evolution of volume fractions, granular dilatancy, and pore-fluid pressure. *Italian Journal of Engineering Geology and Environment*, *43*, 415–424.

Gica, E., Teng, M. H., Liu, P. L.-F., Titov, V., & Zhou, H. (2007). Sensitivity analysis of source parameters for earthquake-generated distant tsunamis. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, *133*(6), 429–441.

Goda, K., Mai, P. M., Yasuda, T., & Mori, N. (2014). Sensitivity of tsunami wave profiles and inundation simulations to earthquake slip and fault geometry for the 2011 tohoku earthquake. *Earth, Planets and Space*, *66*(1), 105.

González, F. I., LeVeque, R. J., Chamberlain, P., Hirai, B., Varkovitzky, J., & George, D. L. (2011). Validation of the geoclaw model. In *Nthmp mms tsunami inundation model validation workshop. geoclaw tsunami modeling group.*

Hanks, T. C., & Kanamori, H. (1979). A moment magnitude scale. *Journal of Geophysical Research: Solid Earth*, *84*(B5), 2348–2350.

Hayes, G. P., Meyers, E. K., Dewey, J. W., Briggs, R. W., Earle, P. S., Benz, H. M., ... others (2017). *Tectonic summaries of magnitude 7 and greater earthquakes from 2000 to 2015* (Tech. Rep.). US Geological Survey.

Hills, J. G., & Mader, C. L. (1997). Tsunami produced by the impacts of small asteroids.

Annals of the New York Academy of Sciences, *822*(1), 381–394.

Johnson, J. M. (1998). Heterogeneous coupling along alaska-aleutians as inferred from tsunami, seismic, and geodetic inversions. In *Advances in geophysics* (Vol. 39, pp. 1–116). Elsevier.

Kajitani, Y., Chang, S. E., & Tatano, H. (2013). Economic impacts of the 2011 tohoku-oki earthquake and tsunami. *Earthquake Spectra*, *29*(s1), S457–S478.

Kanamori, H. (1978). Quantification of earthquakes. *Nature*, *271*(5644), 411.

Kazama, M., & Noda, T. (2012). Damage statistics (summary of the 2011 off the pacific coast of tohoku earthquake damage). *Soils and Foundations*, *52*(5), 780–792.

Kulikov, E. A., Rabinovich, A. B., Thomson, R. E., & Bornhold, B. D. (1996). The landslide tsunami of november 3, 1994, skagway harbor, alaska. *Journal of Geophysical Research: Oceans*, *101*(C3), 6609–6615.

Latter, J. (1981). Tsunamis of volcanic origin: summary of causes, with particular reference to krakatoa, 1883. *Bulletin volcanologique*, *44*(3), 467–490.

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

MacInnes, B. T., Gusman, A. R., LeVeque, R. J., & Tanioka, Y. (2013). Comparison of earthquake source models for the 2011 tohoku event using tsunami simulations and near-field observations. *Bulletin of the Seismological Society of America*, *103*(2B), 1256–1274.

Mader, C. L., & Gittings, M. L. (2002). Modeling the 1958 lituya bay mega-tsunami, ii. *Science of Tsunami Hazards*, *20*(5), 241–250.

Mandli, K. T., & Dawson, C. N. (2014). Adaptive mesh refinement for storm surge. *Ocean Modelling*, *75*, 36–50.

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

Sella, G. F., Dixon, T. H., & Mao, A. (2002). Revel: A model for recent plate velocities from space geodesy. *Journal of Geophysical Research: Solid Earth*, *107*(B4), ETG–11.

Seno, T., Sakurai, T., & Stein, S. (1996). Can the okhotsk plate be discriminated from the north american plate? *Journal of Geophysical Research: Solid Earth*, *101*(B5), 11305–11315.

Shimozono, T., Sato, S., Okayasu, A., Tajima, Y., Fritz, H. M., Liu, H., & Takagawa, T. (2012). Propagation and inundation characteristics of the 2011 tohoku tsunami on the central sanriku coast. *Coastal engineering journal*, *54*(01), 1250004.

Siembieda, W., Johnson, L., & Franco, G. (2012). Rebuild fast but rebuild better: Chile's initial recovery following the 27 february 2010 earthquake and tsunami. *Earthquake Spectra*, *28*(S1), S621–S641.

Yamazaki, Y., & Cheung, K. F. (2011). Shelf resonance and impact of near-field tsunami generated by the 2010 chile earthquake. *Geophysical Research Letters*, *38*(12).

Appendix A - Sub-Fault Parameters

A.1 2010 Chilean Earthquake

Sub-fault geometries used in the above modelling work described above of the 2010 Chilean Mw 8.8 earthquake. Each geometry attempts to use the fewest available number of sub-faults, and correct the slip of each geometry to conserve the total energy of the earthquake. In this instance, cssz refers to the 'Chilean Southern Subduction Zone', as described in Gica (2008).

Table 1: The unconstrained geometry created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Unconstrained Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
csszа89	287.238	-35.5993	3.35	14.52	16.67	11.96	90.0
csszz89	287.7014	-35.6968	3.35	14.52	30.0	26.3	90.0
csszb89	286.7261	-35.4914	3.35	14.52	8.0	5.0	90.0
csszа90	286.8442	-36.5645	3.35	22.64	18.33	11.96	90.0
csszz90	287.2916	-36.7142	3.35	22.64	30.0	27.68	90.0
csszz88	287.9308	-35.2545	3.35	32.81	30.0	24.9	90.0
csszb90	286.3548	-36.4004	3.35	22.64	8.0	5.0	90.0
csszа88	287.5309	-35.0437	3.35	32.81	15.0	11.96	90.0
csszb88	287.0862	-34.8086	3.35	32.81	8.0	5.0	90.0
csszc89	285.91324	-35.31857	3.35	194.52	57.89	4.6	-90.0
csszc90	285.57153	-36.13573	3.35	202.64	57.89	4.6	-90.0
csszz91	287.0726	-37.3224	3.35	10.9	30.0	29.06	90.0
csszа91	286.5925	-37.2488	3.35	10.9	20.0	11.96	90.0
csszb91	286.0721	-37.169	3.35	10.9	8.0	5.0	90.0
csszc88	286.38754	-34.43673	3.35	212.81	57.89	4.6	-90.0
csszа87	288.105	-34.0583	3.35	19.4	15.0	11.96	90.0
csszc91	285.22937	-37.03825	3.35	190.9	57.89	4.6	-90.0
csszb87	287.6115	-33.9142	3.35	19.4	8.0	5.0	90.0
csszc87	286.83475	-33.68572	3.35	199.4	57.89	4.6	-90.0
csszz92	286.9303	-38.152	3.35	8.23	26.67	29.06	90.0
csszа92	286.4254	-38.0945	3.35	8.23	20.0	11.96	90.0

csszb92	285.8948	-38.0341	3.35	8.23	8.0	5.0	90.0
csszc92	285.03525	-37.93476	3.35	188.23	57.89	4.6	-90.0
csszb86	287.8768	-32.9512	3.35	7.01	8.0	5.0	90.0
csszc86	287.06744	-32.86653	3.35	187.01	57.89	4.6	-90.0
cssza86	288.3901	-33.0041	3.35	7.01	15.0	11.96	90.0
cssza93	286.2047	-39.0535	3.35	13.46	20.0	11.96	90.0
csszz93	286.7216	-39.1495	3.35	13.46	23.33	29.06	90.0

Table 2: The single row of along-dip sub-faults created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Along Dip Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
cssza89	287.238	-35.5993	23.47	14.52	16.67	11.96	90.0
csszb89	286.7261	-35.4914	23.47	14.52	8.0	5.0	90.0
csszc89	285.91324	-35.31857	23.47	194.52	57.89	4.6	-90.0
csszz89	287.7014	-35.6968	23.47	14.52	30.0	26.3	90.0

Table 3: The two rows of along-dip sub-faults created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Two Rows of Along Dip Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
cssza89	287.238	-35.5993	11.73	14.52	16.67	11.96	90.0
csszb89	286.7261	-35.4914	11.73	14.52	8.0	5.0	90.0
csszc89	285.91324	-35.31857	11.73	194.52	57.89	4.6	-90.0
csszz89	287.7014	-35.6968	11.73	14.52	30.0	26.3	90.0
cssza90	286.8442	-36.5645	11.73	22.64	18.33	11.96	90.0
csszb90	286.3548	-36.4004	11.73	22.64	8.0	5.0	90.0
csszc90	285.57153	-36.13573	11.73	202.64	57.89	4.6	-90.0
csszz90	287.2916	-36.7142	11.73	22.64	30.0	27.68	90.0

Table 4: The single row of along-strike sub-faults created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Along Strike Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
cssza75	289.6982	-23.1903	3.35	4.83	14.09	11.96	90.0
cssza76	289.6237	-24.0831	3.35	4.67	14.18	11.96	90.0
cssza77	289.5538	-24.9729	3.35	4.3	14.27	11.96	90.0
cssza78	289.4904	-25.8621	3.35	3.86	14.36	11.96	90.0
cssza79	289.3491	-26.8644	3.35	11.34	14.45	11.96	90.0
cssza80	289.1231	-27.7826	3.35	14.16	14.54	11.96	90.0
cssza81	288.8943	-28.6409	3.35	13.19	14.63	11.96	90.0
cssza82	288.7113	-29.468	3.35	9.68	14.72	11.96	90.0
cssza83	288.5944	-30.2923	3.35	5.36	14.81	11.96	90.0
cssza84	288.5223	-31.1639	3.35	3.8	14.9	11.96	90.0
cssza85	288.4748	-32.0416	3.35	2.55	15.0	11.96	90.0
cssza86	288.3901	-33.0041	3.35	7.01	15.0	11.96	90.0
cssza87	288.105	-34.0583	3.35	19.4	15.0	11.96	90.0
cssza88	287.5309	-35.0437	3.35	32.81	15.0	11.96	90.0
cssza89	287.238	-35.5993	3.35	14.52	16.67	11.96	90.0
cssza90	286.8442	-36.5645	3.35	22.64	18.33	11.96	90.0
cssza91	286.5925	-37.2488	3.35	10.9	20.0	11.96	90.0
cssza92	286.4254	-38.0945	3.35	8.23	20.0	11.96	90.0
cssza93	286.2047	-39.0535	3.35	13.46	20.0	11.96	90.0
cssza94	286.0772	-39.7883	3.35	3.4	20.0	11.96	90.0
cssza95	285.9426	-40.776	3.35	9.84	20.0	11.96	90.0

cssza96	285.7839	-41.6303	3.35	7.6	20.0	11.96	90.0
cssza97	285.6695	-42.4882	3.35	5.3	20.0	11.96	90.0
cssza98	285.5035	-43.4553	3.35	10.53	20.0	11.96	90.0
cssza99	285.37	-44.2595	3.35	4.86	20.0	11.96	90.0
cssza100	285.2713	-45.1664	3.35	5.68	20.0	11.96	90.0
cssza101	285.308	-45.8607	3.35	352.58	20.0	9.36	90.0
cssza102	285.2028	-47.1185	3.35	17.72	5.0	9.36	90.0

Table 5: The single row of along-strike sub-faults created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Two Rows of Along Strike Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
cssza82	288.7113	-29.468	4.27	9.68	14.72	11.96	90.0
cssza83	288.5944	-30.2923	4.27	5.36	14.81	11.96	90.0
cssza84	288.5223	-31.1639	4.27	3.8	14.9	11.96	90.0
cssza85	288.4748	-32.0416	4.27	2.55	15.0	11.96	90.0
cssza86	288.3901	-33.0041	4.27	7.01	15.0	11.96	90.0
cssza87	288.105	-34.0583	4.27	19.4	15.0	11.96	90.0
cssza88	287.5309	-35.0437	4.27	32.81	15.0	11.96	90.0
csszz88	287.9308	-35.2545	4.27	32.81	30.0	24.9	90.0
cssza89	287.238	-35.5993	4.27	14.52	16.67	11.96	90.0
csszz89	287.7014	-35.6968	4.27	14.52	30.0	26.3	90.0
cssza90	286.8442	-36.5645	4.27	22.64	18.33	11.96	90.0
csszz90	287.2916	-36.7142	4.27	22.64	30.0	27.68	90.0
cssza91	286.5925	-37.2488	4.27	10.9	20.0	11.96	90.0
csszz91	287.0726	-37.3224	4.27	10.9	30.0	29.06	90.0
cssza92	286.4254	-38.0945	4.27	8.23	20.0	11.96	90.0
csszz92	286.9303	-38.152	4.27	8.23	26.67	29.06	90.0
cssza93	286.2047	-39.0535	4.27	13.46	20.0	11.96	90.0
csszz93	286.7216	-39.1495	4.27	13.46	23.33	29.06	90.0
cssza94	286.0772	-39.7883	4.27	3.4	20.0	11.96	90.0
csszz94	286.6255	-39.8133	4.27	3.4	20.0	29.06	90.0
cssza95	285.9426	-40.776	4.27	9.84	20.0	11.96	90.0

csszz95	286.4921	-40.8481	4.27	9.84	20.0	29.06	90.0

A.2 2011 Tohoku-Oki Earthquake

The following tables are the sub-fault geometries used in the above modelling work described above of the 2011 Tohoku-Oki Mw 9.05 earthquake off the coast of Japan. Each geometry attempts to use the fewest available number of sub-faults, and correct the slip of each geometry to conserve the total energy of the earthquake. The kisz in the name column is represtative of the 'Kamchatka-Yap-Mariana-Izu-Bonin' subduction zones, as described in Gica (2008).

Table 6: The unconstrained geometry created by the automation program that reassigns geometries of the Mw 9.05 Tohoku-Oki earthquake in 2011:

Unconstrained Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
kiszz26	142.2308	38.6421	3.98	188.0	21.0	39.2	90.0
kisza26	142.7622	38.58372	3.98	188.0	21.0	21.28	90.0
kiszz27	142.0269	37.9126	3.98	198.0	21.0	39.2	90.0
kisza27	142.532	37.78298	3.98	198.0	21.0	21.28	90.0
kiszy26	141.699	38.7004	3.98	188.0	21.0	57.12	90.0
kiszy27	141.521	38.0421	3.98	198.0	21.0	57.12	90.0
kiszb26	143.293	38.52536	3.98	188.0	19.0	5.0	90.0
kiszb27	143.0357	37.6534	3.98	198.0	19.0	5.0	90.0
kiszx26	141.1667	38.7588	3.98	188.0	21.0	75.04	90.0
kiszx27	141.0142	38.1717	3.98	198.0	21.0	75.04	90.0
kiszz25	142.3426	39.4907	3.98	185.0	21.0	39.2	90.0
kisza25	142.8839	39.45414	3.98	185.0	21.0	21.28	90.0
kiszz28	141.6671	37.2234	3.98	208.0	21.0	39.2	90.0
kiszy25	141.8012	39.5272	3.98	185.0	21.0	57.12	90.0
kiszy28	141.2016	37.4202	3.98	208.0	21.0	57.12	90.0
kisza28	142.1315	37.02651	3.98	208.0	21.0	21.28	90.0
kiszb25	143.4246	39.41759	3.98	185.0	19.0	5.0	90.0
kiszc26	144.13345	38.43163	3.98	8.0	57.89	4.6	-90.0
kiszc27	143.8325	37.44693	3.98	18.0	57.89	4.6	-90.0
kiszx28	140.7348	37.6171	3.98	208.0	21.0	75.04	90.0
kiszb28	142.5941	36.82965	3.98	208.0	19.0	5.0	90.0

kiszd26	144.43571	38.39835	3.98	8.0	57.89	4.6	-90.0
kiszd27	144.11884	37.37303	3.98	18.0	57.89	4.6	-90.0
kiszc25	144.2811	39.35838	3.98	5.0	57.89	4.6	-90.0
kiszc28	143.32512	36.51672	3.98	28.0	57.89	4.6	-90.0
kiszd25	144.58919	39.33754	3.98	5.0	57.89	4.6	-90.0
kiszz24	142.4312	40.3856	3.98	185.0	21.0	39.2	90.0
kiszz29	141.1506	36.48	3.98	211.0	21.0	39.2	90.0
kisza24	142.9795	40.34903	3.98	185.0	21.0	21.28	90.0
kiszy29	140.7029	36.696	3.98	211.0	21.0	57.12	90.0
kiszy24	141.8827	40.4221	3.98	185.0	21.0	57.12	90.0
kisza29	141.597	36.26404	3.98	211.0	21.0	21.28	90.0
kiszd28	143.58768	36.40445	3.98	28.0	57.89	4.6	-90.0
kiszb24	143.5273	40.31248	3.98	185.0	19.0	5.0	90.0
kiszx24	141.3339	40.4587	3.98	185.0	21.0	75.04	90.0
kiszb29	142.0416	36.04807	3.98	211.0	19.0	5.0	90.0
kiszc24	144.39502	40.25323	3.98	5.0	57.89	4.6	-90.0
kiszd24	144.70712	40.23238	3.98	5.0	57.89	4.6	-90.0
kiszc29	142.74402	35.70493	3.98	31.0	57.89	4.6	-90.0
kiszd29	142.99628	35.58176	3.98	31.0	57.89	4.6	-90.0
kiszb23	143.8028	41.17638	3.98	202.0	19.0	5.0	90.0
kiszz30	140.5883	35.6104	3.98	205.0	21.0	39.2	90.0
kisza30	141.0553	35.4332	3.98	205.0	21.0	21.28	90.0
kisza23	143.2863	41.33346	3.98	202.0	21.0	21.28	90.0
kiszy30	140.1204	35.7876	3.98	205.0	21.0	57.12	90.0
kiszc23	144.61926	40.92623	3.98	22.0	57.89	4.6	-90.0

kiszb30	141.5207	35.25599	3.98	205.0	19.0	5.0	90.0
kiszz23	142.7679	41.4905	3.98	202.0	21.0	39.2	90.0
kiszd23	144.91254	40.83664	3.98	22.0	57.89	4.6	-90.0
kiszy23	142.2482	41.6476	3.98	202.0	21.0	57.12	90.0
kiszc30	142.25653	34.97421	3.98	25.0	57.89	4.6	-90.0
kiszd22	145.1203	41.01272	3.98	62.0	57.89	4.6	-90.0
kiszd30	142.52087	34.87314	3.98	25.0	57.89	4.6	-90.0
kiszx23	141.7273	41.8047	3.98	202.0	21.0	75.04	90.0
kiszc22	144.97125	41.22388	3.98	62.0	57.89	4.6	-90.0
kiszw23	141.205	41.9618	3.98	202.0	21.0	92.95	90.0

Table 7: The single row of along-dip sub-faults created by the automation program that reassigns geometries of the Mw 9.05 Tohoku-Oki earthquake in 2011:

Along Dip Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
kisza26	142.7622	38.58372	31.8	188.0	21.0	21.28	90.0
kiszb26	143.293	38.52536	31.8	188.0	19.0	5.0	90.0
kiszc26	144.13345	38.43163	31.8	8.0	57.89	4.6	-90.0
kiszd26	144.43571	38.39835	31.8	8.0	57.89	4.6	-90.0
kiszx26	141.1667	38.7588	31.8	188.0	21.0	75.04	90.0
kiszy26	141.699	38.7004	31.8	188.0	21.0	57.12	90.0
kiszz26	142.2308	38.6421	31.8	188.0	21.0	39.2	90.0

Table 8: The two rows of along-dip sub-faults created by the automation program that reassigns geometries of the Mw 8.8 Chilean earthquake in 2010:

Two Rows of Along Dip Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
kisza26	142.7622	38.58372	15.9	188.0	21.0	21.28	90.0
kiszb26	143.293	38.52536	15.9	188.0	19.0	5.0	90.0
kiszc26	144.13345	38.43163	15.9	8.0	57.89	4.6	-90.0
kiszd26	144.43571	38.39835	15.9	8.0	57.89	4.6	-90.0
kiszx26	141.1667	38.7588	15.9	188.0	21.0	75.04	90.0
kiszy26	141.699	38.7004	15.9	188.0	21.0	57.12	90.0
kiszz26	142.2308	38.6421	15.9	188.0	21.0	39.2	90.0
kisza27	142.532	37.78298	15.9	198.0	21.0	21.28	90.0
kiszb27	143.0357	37.6534	15.9	198.0	19.0	5.0	90.0
kiszc27	143.8325	37.44693	15.9	18.0	57.89	4.6	-90.0
kiszd27	144.11884	37.37303	15.9	18.0	57.89	4.6	-90.0
kiszx27	141.0142	38.1717	15.9	198.0	21.0	75.04	90.0
kiszy27	141.521	38.0421	15.9	198.0	21.0	57.12	90.0
kiszz27	142.0269	37.9126	15.9	198.0	21.0	39.2	90.0

Table 9: The single row of along-strike sub-faults created by the automation program that reassigns geometries of the Mw 9.05 Tohoku-Oki earthquake in 2011:

Along Strike Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
kiszz0	162.1309	56.4618	6.96	193.84	29.0	50.37	90.0
kiszz1	161.761	55.6033	6.96	195.0	29.0	50.37	90.0
kiszz2	161.3488	54.8127	6.96	200.0	29.0	50.37	90.0
kiszz3	160.8286	54.0312	6.96	204.0	29.0	50.37	90.0
kiszz4	160.2246	53.3051	6.96	210.0	29.0	50.37	90.0
kiszz5	159.5122	52.6531	6.96	218.0	29.0	50.37	90.0
kiszz6	158.6263	51.9452	6.96	218.0	29.0	50.37	90.0
kiszz7	157.7443	51.1745	6.96	214.0	29.0	50.37	90.0
kiszz8	156.9956	50.4829	6.96	218.0	31.0	53.45	90.0
kiszz9	156.1556	49.8058	6.96	220.0	31.0	53.45	90.0
kiszz10	155.2865	49.133	6.96	221.0	31.0	53.45	90.0
kiszz11	154.3991	48.4244	6.96	219.0	31.0	53.45	90.0
kiszz12	153.5435	47.7046	6.96	217.0	31.0	53.45	90.0
kiszz13	152.7801	46.9934	6.96	218.0	31.0	53.45	90.0
kiszz14	151.9426	46.4438	6.96	225.0	23.0	44.08	90.0
kiszz15	151.1151	45.9214	6.96	233.0	25.0	44.86	90.0
kiszz16	150.1422	45.439	6.96	237.0	25.0	44.86	90.0
kiszz17	149.0865	44.9498	6.96	237.0	25.0	44.86	90.0
kiszz18	148.0194	44.4316	6.96	235.0	25.0	44.86	90.0
kiszz19	146.9872	43.887	6.96	233.0	25.0	44.86	90.0
kiszz20	146.047	43.4047	6.96	237.0	25.0	44.86	90.0

kiszz21	145.0475	42.9438	6.96	239.0	25.0	44.86	90.0
kiszz22	144.0455	42.5225	6.96	242.0	25.0	44.86	90.0
kiszz23	142.7679	41.4905	6.96	202.0	21.0	39.2	90.0
kiszz24	142.4312	40.3856	6.96	185.0	21.0	39.2	90.0
kiszz25	142.3426	39.4907	6.96	185.0	21.0	39.2	90.0
kiszz26	142.2308	38.6421	6.96	188.0	21.0	39.2	90.0
kiszz27	142.0269	37.9126	6.96	198.0	21.0	39.2	90.0
kiszz28	141.6671	37.2234	6.96	208.0	21.0	39.2	90.0
kiszz29	141.1506	36.48	6.96	211.0	21.0	39.2	90.0
kiszz30	140.5883	35.6104	6.96	205.0	21.0	39.2	90.0
kiszz31	140.1979	34.5512	6.96	190.0	22.0	40.83	90.0

Table 10: The single row of along-strike sub-faults created by the automation program that reassigns geometries of the Mw 9.05 Tohoku-Oki earthquake in 2011:

Two Rows of Along Strike Geometry							
Name	Longitude	Latitude	Slip(m)	Strike(deg)	Dip(deg)	Depth(km)	Rake(deg)
kiszz12	153.5435	47.7046	4.64	217.0	31.0	53.45	90.0
kisza12	153.9994	47.47286	4.64	217.0	31.0	27.7	90.0
kiszz13	152.7801	46.9934	4.64	218.0	31.0	53.45	90.0
kisza13	153.2239	46.75639	4.64	218.0	31.0	27.7	90.0
kiszz14	151.9426	46.4438	4.64	225.0	23.0	44.08	90.0
kisza14	152.3657	46.15145	4.64	225.0	23.0	24.54	90.0
kiszz15	151.1151	45.9214	4.64	233.0	25.0	44.86	90.0
kisza15	151.4663	45.59629	4.64	233.0	25.0	23.73	90.0
kiszz16	150.1422	45.439	4.64	237.0	25.0	44.86	90.0
kisza16	150.4572	45.09765	4.64	237.0	25.0	23.73	90.0
kiszz17	149.0865	44.9498	4.64	237.0	25.0	44.86	90.0
kisza17	149.3989	44.60839	4.64	237.0	25.0	23.73	90.0
kiszz18	148.0194	44.4316	4.64	235.0	25.0	44.86	90.0
kisza18	148.3454	44.09819	4.64	235.0	25.0	23.73	90.0
kiszz19	146.9872	43.887	4.64	233.0	25.0	44.86	90.0
kisza19	147.3262	43.5619	4.64	233.0	25.0	23.73	90.0
kiszz20	146.047	43.4047	4.64	237.0	25.0	44.86	90.0
kisza20	146.3513	43.06326	4.64	237.0	25.0	23.73	90.0
kiszz21	145.0475	42.9438	4.64	239.0	25.0	44.86	90.0
kisza21	145.3331	42.59483	4.64	239.0	25.0	23.73	90.0
kiszz22	144.0455	42.5225	4.64	242.0	25.0	44.86	90.0

kisza22	144.3041	42.16312	4.64	242.0	25.0	23.73	90.0
kiszz23	142.7679	41.4905	4.64	202.0	21.0	39.2	90.0
kisza23	143.2863	41.33346	4.64	202.0	21.0	21.28	90.0
kiszz24	142.4312	40.3856	4.64	185.0	21.0	39.2	90.0
kisza24	142.9795	40.34903	4.64	185.0	21.0	21.28	90.0
kiszz25	142.3426	39.4907	4.64	185.0	21.0	39.2	90.0
kisza25	142.8839	39.45414	4.64	185.0	21.0	21.28	90.0
kiszz26	142.2308	38.6421	4.64	188.0	21.0	39.2	90.0
kisza26	142.7622	38.58372	4.64	188.0	21.0	21.28	90.0
kiszz27	142.0269	37.9126	4.64	198.0	21.0	39.2	90.0
kisza27	142.532	37.78298	4.64	198.0	21.0	21.28	90.0
kiszz28	141.6671	37.2234	4.64	208.0	21.0	39.2	90.0
kisza28	142.1315	37.02651	4.64	208.0	21.0	21.28	90.0
kiszz29	141.1506	36.48	4.64	211.0	21.0	39.2	90.0
kisza29	141.597	36.26404	4.64	211.0	21.0	21.28	90.0
kiszz30	140.5883	35.6104	4.64	205.0	21.0	39.2	90.0
kisza30	141.0553	35.4332	4.64	205.0	21.0	21.28	90.0
kiszz31	140.1979	34.5512	4.64	190.0	22.0	40.83	90.0
kisza31	140.6956	34.4789	4.64	190.0	22.0	22.1	90.0
kisza32	141.0551	33.0921	4.64	180.0	32.0	23.48	90.0
kisza33	141.0924	32.1047	4.64	173.85	27.65	20.67	90.0
kisza34	141.1869	31.1851	4.64	172.14	25.0	18.26	90.0
kisza35	141.4154	30.1707	4.64	162.98	25.0	17.12	90.0
kisza36	141.6261	29.274	4.64	161.68	25.73	18.71	90.0
kisza37	142.012	28.3322	4.64	154.72	20.0	14.54	90.0

kisza38	142.2254	27.6946	4.64	170.27	20.0	14.54	90.0
kisza39	142.3085	26.9127	4.64	177.23	24.23	17.42	90.0

Appendix B - Wave Gauge Locations

Table 11: The locations of the 18 Wave Gauges implemented in GeoClaw:

Number	Latitude (^{o}N)	Longitude (^{o}E)	Location Description
1	21.89638000	200.4076800	Hanapepe Harbor, HI
2	21.29147000	202.1262300	Honolulu Harbor, HI
3	21.72998000	202.0100500	North West of Kawela Bay, HI
4	21.00330000	203.0411700	Between Moloka'i and Lana'i Islands, HI
5	20.78462000	203.0075800	South West of Lana'i Island, HI
6	21.00178000	203.3081800	North West of Kapalua, HI
7	20.86727000	203.2528300	Between the Islands of Lana'i and Maui, HI
8	20.86845000	203.3147200	South West of Lahaina, HI
9	20.61252000	203.4918800	South West of Molokini, HI
10	20.76525000	203.5076500	South East of Maalaea, HI
11	20.90213000	203.5282500	North of Kahului, HI
12	20.76123000	204.0219000	Hana Bay, HI
13	19.74170000	204.9180200	Hilo Bay, HI
14	19.74167000	204.9300300	Hilo Bay Behind Breakwater, HI
15	20.26502000	204.0984300	North West of Kapaau, HI
16	20.04108000	204.1583700	North West of Kawaihai, HI
17	19.63510000	204.0008500	Kailua-Kona Harbor, HI
18	19.66780000	203.9681000	North West of Kailua-Kona, HI

Appendix C - Python Codes

C.1 Download Bathymetry

This Python script downloads bathymetry from NOAA of both sides of the anti-meridian, and combines them into a single file.

```
1  from __future__ import absolute_import
2  from __future__ import print_function
3  import os
4  import numpy as np
5  import clawpack.clawutil.data
6
7  try:
8      CLAW = os.environ['CLAW']
9  except:
10     raise Exception("*** Must first set CLAW enviornment variable")
11 #Below condition is an example to download etopo1 data for longitude
       ↪ (130-215) &
12 # latitude(0-50) in 2min interval.
13 #Directory
       ↪ ------------------------------------------------------------------#
       ↪
14 #scratch_dir: directory to save files
15 # scratch_dir = '/Users/wiersma/work/geoclaw'
16 scratch_dir = '/Users/wiersma/work/geoclaw'
17 #topo_path1: 1st file path & name to append
18 topo_path2 = os.path.join(scratch_dir, 'etopo1_-180_-50_-60_60_2min.asc')
19 #topo_path2: 2nd file path & name to append
```

```
20 topo_path1 = os.path.join(scratch_dir, 'etopo1_120_180_-60_60_2min.asc')
21 #result_path1: final topo file path & name
22 result_path1 = os.path.join(scratch_dir, 'geoclawtopofile.asc')
23 #
    ↪ ------------------------------------------------------------------------#
    ↪
24 def make_topo():
25     from clawpack.geoclaw import etopotools
26     from clawpack.geoclaw import topotools
27     topopath=scratch_dir
28     etopotools.etopo1_download((120, 180), (-60, 60), dx=0.0333333333,
        ↪ output_dir=topopath)
29     etopotools.etopo1_download((-180, -50), (-60, 60), dx=0.0333333333,
        ↪ output_dir=topopath)
30
31
32     # combining the two downloaded files into a single file
33 def append_topo():
34
35     with open('etopo1_120_180_-60_60_2min.asc') as file:
36         next(file)
37         next(file)
38         next(file)
39         next(file)
40         next(file)
```

```
41          next(file)
42          tf1rows = [[float(digit) for digit in line.split()] for line in file
                ↪ ]
43
44      with open('etopo1_-180_-50_-60_60_2min.asc') as file:
45          next(file)
46          next(file)
47          next(file)
48          next(file)
49          next(file)
50          next(file)
51          tf2rows = [[float(digit) for digit in line.split()] for line in file
                ↪ ]
52
53      tf1=[]
54      tf2=[]
55
56      tf1=np.asarray(tf1rows)
57      tf2=np.asarray(tf2rows)
58      tf3=[]
59      tf4=[]
60      tf3=np.delete(tf1, -1, -1)
61      tf4=np.concatenate((tf3,tf2), axis=1)
62
63
```

```
64 # create the header for topofile
65     header = "ncols␣␣␣5701\n"
66     header += "nrows␣␣3601\n"
67     header += "xllcorner␣␣␣␣120.0\n"
68     header += "yllcorner␣␣␣␣-60.0\n"
69     header +="cellsize␣␣␣␣␣0.033333333300\n"
70     header += "nodata_value␣-9999\n"
71 
72     with open("geoclawtopofile.asc", "w") as f:
73         f.write(header)
74         np.savetxt(f, tf4, fmt="%4.2f")
75     print("third", np.shape(tf4))
76 
77 if __name__=='__main__':
78     make_topo()
79     result= append_topo()
```

C.2 Geometry Variation

This Python script creates the different geometries through either the Haversine equation or through the manipulation of sub-fault names provided by NOAA's SIFT database. It should be noted that this study does not use the SIFT program, merely the sub-fault system.

```
1  import csv
2  import urllib2
3  import numpy as np
4  from numpy import argsort
5  from math import radians, cos, sin, asin, sqrt, ceil, floor
6  from mpl_toolkits.basemap import Basemap
7  import matplotlib.pyplot as plt
8  import pylab
9  from variables import eq_lat, eq_lon, eq_magnitude
10
11 ###NOAA SIFT Sub-Faults are located in "locations.txt"
12 name=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(0),
       ↪ dtype='str')
13 lon=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(2))
14 lat=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(3))
15 strike=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(5))
16 dip=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(6))
17 depth=np.loadtxt('locations.txt', delimiter=',␣', skiprows=1, usecols=(7))
18
19 ####Haversine calculator
```

```
20 def gc_dist(lon1, lat1, lon2, lat2):
21     """
22     Calculate the great circle distance between two points
23     on the earth (specified in decimal degrees)
24     """
25     # convert decimal degrees to radians
26     lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
27 
28     # haversine formula
29     dlon = lon2 - lon1
30     dlat = lat2 - lat1
31     a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
32     c = 2 * asin(sqrt(a))
33     r = 6371 # Radius of earth in kilometers. Use 3956 for miles
34     dist=c*r
35     return dist
36 
37 #find the distance of all of the points throughout the arrays lon and lat
38 distances=[]
39 for i in xrange(len(lon)):
40     iteration=gc_dist(eq_lon,eq_lat,lon[i],lat[i])
41     distances.append(iteration)
42 distances = np.asarray(distances)
43 
44 #find closest indicies to epicenter
```

```
45 dummy = np.linspace(0,len(distances)-1,len(distances),dtype=int)
46 inds = distances.argsort()
47 closestfaults = dummy[inds]
48
49 #find the surrounding faults nearest the closest fault to epicenter
50 faultdistances=[]
51 for i in xrange(len(lon)):
52     iterationfaults=gc_dist(lon[closestfaults[0]], lat[closestfaults[0]],
       ↪ lon[i],lat[i])
53     faultdistances.append(iterationfaults)
54
55
56 mw=eq_magnitude
57 unit_source_area = 50e3*100e3
58
59 #find the required number of sources, with the minimum being one
60 area=(10**((mw-4.441)/0.846))*1e6
61 if floor(area/unit_source_area) > 1.0:
62     num_sources=(floor(area/unit_source_area))
63 else:
64     num_sources=1.0
65
66 moment=(10**(1.5*(mw+6.07)))
67 slip=(moment/(4.3e10*area))
68
```

```
69 #changing geometries
70 #make it wide
71 wnames=np.linspace(0,num_sources,num_sources+1)
72 wide_names=np.zeros(int(num_sources))
73 wide_geo=np.zeros(int(num_sources))
74 wide_names[0]=(int(name[closestfaults[0]][5:8])-(int(0.5*num_sources)))
75 widecsv=open("widegeom.txt", "w")
76 widecsv.write ('Name,␣Filename/URL,␣Long(deg),␣Lat(deg),␣Slip(m),␣Strike(deg
      ↪ ),␣Dip(deg),␣Depth(km),␣Length(km),␣Width(km),␣Rake(deg)')
77 widecsv.write('\n')
78 for i in xrange(int(num_sources)):
79     wide_names[i]=(wide_names[0]+wnames[i])
80     wide_geo=(name[closestfaults[0]][0:5])+str(int(wide_names[i]))
81     with open('locations.txt') as infile:
82         for line in infile:
83             if wide_geo in line:
84                 widecsv.write('␣'.join(line.split()))
85                 widecsv.write('\n')
86                 break
87
88 wide_sources=len(wide_names)
89 wideslip=(moment/(4.3e10*(wide_sources*5.e9)))
90
91 #2 wide
92 w2geom=open("2widegeom.txt","w")
```

```
93  w2geom.write ('Name,␣Filename/URL,␣Long(deg),␣Lat(deg),␣Slip(m),␣Strike(deg)
        ↪ ,␣Dip(deg),␣Depth(km),␣Length(km),␣Width(km),␣Rake(deg)')
94  w2geom.write ('\n')
95  w2names=np.linspace(0,0.5*num_sources,int(0.5*num_sources)+1)
96  wide_names21=np.zeros(int(0.5*num_sources))
97  wide_names22=np.zeros(int(0.5*num_sources))
98  wide_geom21=np.zeros(int(0.5*num_sources))
99  wide_geom22=np.zeros(int(0.5*num_sources))
100 wide_names21[0]=(int(name[closestfaults[0]][5:8])-(int(0.25*num_sources)))
101 wide_names22[0]=(int(name[closestfaults[0]][5:8])-(int(0.25*num_sources)))
102 nextletter=np.zeros(int(0.5*num_sources))
103 for i in xrange (int(0.5*num_sources)):
104     wide_names21[i]=(wide_names21[0]+w2names[i])
105     wide_geom21=(name[closestfaults[0]][0:5])+str(int(wide_names21[i]))
106     if (name[closestfaults[0]][4:5]) != (name[closestfaults[1]][4:5]):
107         nextletter=(name[closestfaults[1]][4:5])
108     elif (name[closestfaults[1]][4:5]) != (name[closestfaults[2]][4:5]):
109         nextletter=(name[closestfaults[2]][4:5])
110     elif (name[closestfaults[2]][4:5]) != (name[closestfaults[3]][4:5]):
111         nextletter=(name[closestfaults[3]][4:5])
112     wide_names22[i]=(wide_names22[0]+w2names[i])
113     wide_geom22=(str(name[closestfaults[0]][0:4])+(str(nextletter))+str(int(
        ↪ wide_names22[i])))
114     with open('locations.txt') as infile:
115         for line in infile:
```

```
116             if wide_geom21 in line:
117                 w2geom.write('␣␣␣'.join(line.split()))
118                 w2geom.write('\n')
119                 break
120     with open('locations.txt') as infile:
121         for line in infile:
122             if wide_geom22 in line:
123                 w2geom.write('␣␣␣'.join(line.split()))
124                 w2geom.write('\n')
125                 break
126 w2geom.close()
127 wide2_sources=((int(len(wide_names21)))+(int(len(wide_names22))))
128 wide2slip=(moment/(4.3e10*(wide2_sources*5.e9)))
129
130 #blob
131 blob=open("amorphous.txt","w")
132 blob.write ('Name,␣Filename/URL,␣Long(deg),␣Lat(deg),␣Slip(m),␣Strike(deg),␣
        ↪ Dip(deg),␣Depth(km),␣Length(km),␣Width(km),␣Rake(deg)\n')
133 # blob.write ('\n')
134 for i in xrange(int(num_sources)):
135     with open('locations.txt') as infile:
136         for line in infile:
137             if (name[closestfaults[i]][:]) in line:
138                 blob.write('␣␣␣␣'.join(line.split()))
139                 blob.write('\n')
```

```
#long
lgeom=open("longgeom.txt","w")
lgeom2=open("2longgeom.txt","w")
lgeom.write ('Name, Filename/URL, Long(deg), Lat(deg), Slip(m), Strike(deg),
    ↪ Dip(deg), Depth(km), Length(km), Width(km), Rake(deg)')
lgeom2.write (str('Name, Filename/URL, Long(deg), Lat(deg), Slip(m), Strike(
    ↪ deg), Dip(deg), Depth(km), Length(km), Width(km), Rake(deg)'))
lgeom.write ('\n')
lgeom2.write ('\n')
def makelong(random_input):
    long_names=[]
    long_geo=np.zeros(int(num_sources))
    longvariable=list((name[closestfaults[0]][0:-1]))
    longstart=name[closestfaults[0]][0:4]
    longstartnumber=name[closestfaults[0]][5:7]
    long_names2=[]
    for i in xrange(int(len(letter))):
        if (str(longstart)+str(letter[i])+str(longstartnumber)) in open('
            ↪ locations.txt').read():
            long_names.append(str(longstart)+str(letter[i])+str(
                ↪ longstartnumber))
            with open('locations.txt') as infile:
                for line in infile:
                    if (str(longstart)+str(letter[i])+str(longstartnumber)) in
                        ↪  line:
```

```
160                     lgeom.write('␣␣␣'.join(line.split()))
161                     lgeom.write('\n')
162                     break
163     for i in xrange (len(letter)):
164         if (str(longstart)+str(letter[i])+str(longstartnumber)) in open('
              ↪ locations.txt').read():
165             long_names2.append(str(longstart)+str("a")+str(longstartnumber))
166             with open('locations.txt') as infile:
167                 for line in infile:
168                     if (str(longstart)+str(letter[i])+str(longstartnumber)) in
                          ↪  line:
169                         lgeom2.write('␣␣␣'.join(line.split()))
170                         lgeom2.write('\n')
171                         break
172     for i in xrange (len(letter)):
173         if (str(longstart)+str(letter[i])+str(int(longstartnumber)+1)) in
              ↪ open('locations.txt').read():
174             long_names2.append(str(longstart)+str(letter[i])+str(int(
                  ↪ longstartnumber)+1))
175             with open('locations.txt') as infile:
176                 for line in infile:
177                     if (str(longstart)+str(letter[i])+str(int(longstartnumber)
                          ↪ +1)) in line:
178                         lgeom2.write('␣␣␣'.join(line.split()))
179                         lgeom2.write('\n')
```

```
                    break
    return long_names, long_names2
letter=['a', 'b', 'c', 'd', 'w', 'x', 'y', 'z']
long_names, long_names2=makelong(letter)
lgeom.close()
long_sources=len(long_names)
longslip=(moment/(4.3e10*(long_sources*5.e9)))
lgeom2.close()
long2_sources=len(long_names2)
long2slip=(moment/(4.3e10*(long2_sources*5.e9)))
```

C.3 Modified GeoClaw Files

This section contains the modified GeoClaw files. When GeoClaw is first installed, the programs need to be modified to contain a new domain of calculations, a new domain of plots, and modify to read sub-fault names used by NOAA rather than implementing fault parameters manually.

maketopo.py

```
"""
Create topo and dtopo files needed for this example:
    etopo10min120W60W60S0S.asc download from GeoClaw topo repository
    dtopo_usgs100227.tt3 create using Okada model
Prior to Clawpack 5.2.1, the fault parameters we specified in a .cfg file,
but now they are explicit below.

Call functions with makeplots==True to create plots of topo, slip, and
    ↪ dtopo.
"""

from __future__ import absolute_import
from __future__ import print_function
import os

import clawpack.clawutil.data

```

```
17 try:
18     CLAW = os.environ['CLAW']
19 except:
20     raise Exception("*** Must first set CLAW enviornment variable")
21
22 # # Scratch directory for storing topo and dtopo files:
23 # scratch_dir = os.path.join(CLAW, 'geoclaw', 'scratch')
24 #
25 # def get_topo(makeplots=False):
26 # """
27 # Retrieve the topo file from the GeoClaw repository.
28 # """
29 # from clawpack.geoclaw import topotools
30 # topo_fname = 'etopo10min120W60W60S0S.asc'
31 # url = 'http://www.geoclaw.org/topo/etopo/' + topo_fname
32 # clawpack.clawutil.data.get_remote_file(url, output_dir=scratch_dir,
33 # file_name=topo_fname, verbose=True)
34 #
35 # if makeplots:
36 # from matplotlib import pyplot as plt
37 # topo = topotools.Topography(os.path.join(scratch_dir,topo_fname),
     ↪ topo_type=2)
38 # topo.plot()
39 # fname = os.path.splitext(topo_fname)[0] + '.png'
40 # plt.savefig(fname)
```

```
41 # print("Created ",fname)
42
43 class Maketopo:
44     def __init__(self, sift_slip, eq_name):
45         self.slip = sift_slip
46         self.eq_name = eq_name
47         #self.lat = lat
48         #self.lon = lon
49         #self.depth = depth
50         #self.strike = strike
51         #self.dip = dip
52         #self.length = length
53         #self.width = width
54         #self.index = i
55         #self.slip = slip
56         #self.rake = rake
57
58
59     def make_dtopo(self, makeplots=False):
60         """
61         Create dtopo data file for deformation of sea floor due to
               ↪ earthquake.
62         Uses the Okada model with fault parameters and mesh specified below
               ↪ .
63         """
```

```
64        from clawpack.geoclaw import dtopotools
65        import numpy
66
67        dtopo_fname = "dtopo_{}.tt3".format(self.eq_name)# % self.index
68
69        # Specify subfault parameters for this simple fault model
             ↪ consisting
70        # of a single subfault:
71
72        #usgs_subfault = dtopotools.SubFault()
73        #usgs_subfault.strike = self.strike
74        ##usgs_subfault.length = self.length*1e3
75        #usgs_subfault.width = self.width*1e3
76        #usgs_subfault.depth = self.depth*1e3
77        #usgs_subfault.slip = self.slip
78        #usgs_subfault.rake = self.rake
79        #usgs_subfault.dip = self.dip
80        #usgs_subfault.longitude = self.lon
81        #usgs_subfault.latitude = self.lat
82        #usgs_subfault.coordinate_specification = "centroid"
83        print(self.slip)
84        fault = dtopotools.SiftFault(self.slip)
85        #fault.subfaults = [usgs_subfault]
86        #
87        # print("Mw = ",fault.Mw())
```

```
88          # f = open("Mw_%03d.txt" %self.index, 'w')
89          # f.write(str(fault.Mw()))
90          # f.close()
91
92
93          if os.path.exists(dtopo_fname):
94              print("*** Not regenerating dtopo file (already exists): %s" \
95                          % dtopo_fname)
96          else:
97              print("Using Okada model to create dtopo file")
98              """
99              # Lower and upper edge of computational domain:
100             clawdata.lower[0] = 121. # west longitude
101             clawdata.upper[0] = 290. # east longitude
102
103             clawdata.lower[1] = -59. # south latitude
104             clawdata.upper[1] = 59. # north latitude
105             """
106             x = numpy.linspace(121, 310, 100)
107             y = numpy.linspace(-59, 59, 100)
108             times = [1.]
109
110             fault.create_dtopography(x,y,times)
111             dtopo = fault.dtopo
112             dtopo.write(dtopo_fname, dtopo_type=3)
```

```
113
114
115         if makeplots:
116             from matplotlib import pyplot as plt
117             if fault.dtopo is None:
118                 # read in the pre-existing file:
119                 print("Reading␣in␣dtopo␣file...")
120                 dtopo = dtopotools.DTopography()
121                 dtopo.read(dtopo_fname, dtopo_type=3)
122                 x = dtopo.x
123                 y = dtopo.y
124             plt.figure(figsize=(12,7))
125             ax1 = plt.subplot(121)
126             ax2 = plt.subplot(122)
127             fault.plot_subfaults(axes=ax1,slip_color=True)
128             ax1.set_xlim(x.min(),x.max())
129             ax1.set_ylim(y.min(),y.max())
130             dtopo.plot_dZ_colors(1.,axes=ax2)
131             fname = os.path.splitext(os.path.split(dtopo_fname)[-1])[0] + '.
                    ↪ png'
132             plt.savefig(fname)
133             print("Created␣",fname)
134
135
136 # if __name__=='__main__':
```

```
137 # mtopo = Maketopo({}, False)
138 # mtopo.make_dtopo(False)
```

C.3.1 setrun.py

```
"""
Module to set up run time parameters for Clawpack.

The values set in the function setrun are then written out to data files
that will be read in by the Fortran code.

"""

from __future__ import absolute_import
from __future__ import print_function
import os
import numpy as np



try:
    CLAW = os.environ['CLAW']
except:
    raise Exception("*** Must first set CLAW enviornment variable")

# Scratch directory for storing topo and dtopo files:
#scratch_dir = os.path.join(CLAW, 'geoclaw', 'scratch')
scratch_dir = '/Users/wiersma/work/geoclaw'

```

```
25 #------------------------------
26 def setrun(claw_pkg='geoclaw'):
27 #------------------------------
28
29     """
30     Define the parameters used for running Clawpack.
31
32     INPUT:
33         claw_pkg expected to be "geoclaw" for this setrun.
34
35     OUTPUT:
36         rundata - object of class ClawRunData
37
38     """
39
40     from clawpack.clawutil import data
41
42     assert claw_pkg.lower() == 'geoclaw', "Expected␣claw_pkg␣=␣'geoclaw'"
43
44     num_dim = 2
45     rundata = data.ClawRunData(claw_pkg, num_dim)
46
47
48     #------------------------------------------------------------------
49     # Problem-specific parameters to be written to setprob.data:
```

```
50    #------------------------------------------------------------------
51
52    #probdata = rundata.new_UserData(name='probdata',fname='setprob.data')
53
54
55    #------------------------------------------------------------------
56    # GeoClaw specific parameters:
57    #------------------------------------------------------------------
58    rundata = setgeo(rundata)
59
60    #------------------------------------------------------------------
61    # Standard Clawpack parameters to be written to claw.data:
62    # (or to amr2ez.data for AMR)
63    #------------------------------------------------------------------
64    clawdata = rundata.clawdata # initialized when rundata instantiated
65
66
67    # Set single grid parameters first.
68    # See below for AMR parameters.
69
70
71    # ---------------
72    # Spatial domain:
73    # ---------------
74
```

```
75      # Number of space dimensions:
76      clawdata.num_dim = num_dim
77
78      # Lower and upper edge of computational domain:
79      clawdata.lower[0] = 121. # west longitude
80      clawdata.upper[0] = 309.0 # east longitude
81
82      clawdata.lower[1] = -59.0 # south latitude
83      clawdata.upper[1] = 59.0 # north latitude
84
85
86
87      # Number of grid cells: Coarsest grid
88      clawdata.num_cells[0] = 60
89      clawdata.num_cells[1] = 60
90
91      # ---------------
92      # Size of system:
93      # ---------------
94
95      # Number of equations in the system:
96      clawdata.num_eqn = 3
97
98      # Number of auxiliary variables in the aux array (initialized in setaux
           ↪ )
```

```
99    clawdata.num_aux = 3
100
101   # Index of aux array corresponding to capacity function, if there is
          ↪ one:
102   clawdata.capa_index = 2
103
104
105
106   # -------------
107   # Initial time:
108   # -------------
109
110   clawdata.t0 = 30000.0
111
112
113   # Restart from checkpoint file of a previous run?
114   # If restarting, t0 above should be from original run, and the
115   # restart_file 'fort.chkNNNNN' specified below should be in
116   # the OUTDIR indicated in Makefile.
117
118   clawdata.restart = False # True to restart from prior results
119   clawdata.restart_file = 'fort.chk00096' # File to use for restart data
120
121   # -------------
122   # Output times:
```

```
123     #--------------
124
125     # Specify at what times the results should be written to fort.q files.
126     # Note that the time integration stops after the final output time.
127     # The solution at initial time t0 is always written in addition.
128
129     clawdata.output_style = 1
130
131     if clawdata.output_style==1:
132         # Output nout frames at equally spaced times up to tfinal:
133         clawdata.num_output_times = 250
134         # clawdata.tfinal = 9*3600.
135         clawdata.tfinal = 100000.
136         clawdata.output_t0 = True # output at initial (or restart) time?
137
138     elif clawdata.output_style == 2:
139         # Specify a list of output times.
140         clawdata.output_times = [0.5, 1.0]
141
142     elif clawdata.output_style == 3:
143         # Output every iout timesteps with a total of ntot time steps:
144         clawdata.output_step_interval = 1
145         clawdata.total_steps = 3
146         clawdata.output_t0 = True
147
```

```
148
149     clawdata.output_format = 'ascii' # 'ascii' or 'netcdf'
150
151     clawdata.output_q_components = 'all' # need all
152     clawdata.output_aux_components = 'none' # eta=h+B is in q
153     clawdata.output_aux_onlyonce = False # output aux arrays each frame
154
155
156
157     # --------------------------------------------------
158     # Verbosity of messages to screen during integration:
159     # --------------------------------------------------
160
161     # The current t, dt, and cfl will be printed every time step
162     # at AMR levels <= verbosity. Set verbosity = 0 for no printing.
163     # (E.g. verbosity == 2 means print only on levels 1 and 2.)
164     clawdata.verbosity = 1
165
166
167
168     # --------------
169     # Time stepping:
170     # --------------
171
172     # if dt_variable==1: variable time steps used based on cfl_desired,
```

```
173     # if dt_variable==0: fixed time steps dt = dt_initial will always be
            ↪ used.
174     clawdata.dt_variable = True
175
176     # Initial time step for variable dt.
177     # If dt_variable==0 then dt=dt_initial for all steps:
178     clawdata.dt_initial = 0.2
179
180     # Max time step to be allowed if variable dt used:
181     clawdata.dt_max = 1e+99
182
183     # Desired Courant number if variable dt used, and max to allow without
184     # retaking step with a smaller dt:
185     clawdata.cfl_desired = 0.75
186     clawdata.cfl_max = 1.0
187
188     # Maximum number of time steps to allow between output times:
189     clawdata.steps_max = 5000
190
191
192
193
194     # ------------------
195     # Method to be used:
196     # ------------------
```

```
197
198     # Order of accuracy: 1 => Godunov, 2 => Lax-Wendroff plus limiters
199     clawdata.order = 2
200
201     # Use dimensional splitting? (not yet available for AMR)
202     clawdata.dimensional_split = 'unsplit'
203
204     # For unsplit method, transverse_waves can be
205     # 0 or 'none' ==> donor cell (only normal solver used)
206     # 1 or 'increment' ==> corner transport of waves
207     # 2 or 'all' ==> corner transport of 2nd order corrections too
208     clawdata.transverse_waves = 2
209
210     # Number of waves in the Riemann solution:
211     clawdata.num_waves = 3
212
213     # List of limiters to use for each wave family:
214     # Required: len(limiter) == num_waves
215     # Some options:
216     # 0 or 'none' ==> no limiter (Lax-Wendroff)
217     # 1 or 'minmod' ==> minmod
218     # 2 or 'superbee' ==> superbee
219     # 3 or 'mc' ==> MC limiter
220     # 4 or 'vanleer' ==> van Leer
221     clawdata.limiter = ['mc', 'mc', 'mc']
```

```
222
223     clawdata.use_fwaves = True # True ==> use f-wave version of algorithms
224
225     # Source terms splitting:
226     # src_split == 0 or 'none' ==> no source term (src routine never called
            ↪ )
227     # src_split == 1 or 'godunov' ==> Godunov (1st order) splitting used,
228     # src_split == 2 or 'strang' ==> Strang (2nd order) splitting used, not
            ↪  recommended.
229     clawdata.source_split = 'godunov'
230
231
232     # --------------------
233     # Boundary conditions:
234     # --------------------
235
236     # Number of ghost cells (usually 2)
237     clawdata.num_ghost = 2
238
239     # Choice of BCs at xlower and xupper:
240     # 0 => user specified (must modify bcN.f to use this option)
241     # 1 => extrapolation (non-reflecting outflow)
242     # 2 => periodic (must specify this at both boundaries)
243     # 3 => solid wall for systems where q(2) is normal velocity
244
```

```
245     clawdata.bc_lower[0] = 'extrap'
246     clawdata.bc_upper[0] = 'extrap'
247
248     clawdata.bc_lower[1] = 'extrap'
249     clawdata.bc_upper[1] = 'extrap'
250
251
252
253     # --------------
254     # Checkpointing:
255     # --------------
256
257     # Specify when checkpoint files should be created that can be
258     # used to restart a computation.
259
260     clawdata.checkpt_style = 0
261
262     if clawdata.checkpt_style == 0:
263         # Do not checkpoint at all
264         pass
265
266     elif clawdata.checkpt_style == 1:
267         # Checkpoint only at tfinal.
268         pass
269
```

```
270     elif clawdata.checkpt_style == 2:
271         # Specify a list of checkpoint times.
272         clawdata.checkpt_times = [0.1,0.15]
273 
274     elif clawdata.checkpt_style == 3:
275         # Checkpoint every checkpt_interval timesteps (on Level 1)
276         # and at the final time.
277         clawdata.checkpt_interval = 5
278 
279 
280     # ---------------
281     # AMR parameters:
282     # ---------------
283     amrdata = rundata.amrdata
284 
285     # max number of refinement levels:
286     amrdata.amr_levels_max = 3
287 
288     # List of refinement ratios at each level (length at least mxnest-1)
289     amrdata.refinement_ratios_x = [2,6]
290     amrdata.refinement_ratios_y = [2,6]
291     amrdata.refinement_ratios_t = [2,6]
292 
293 
294     # Specify type of each aux variable in amrdata.auxtype.
```

```
295     # This must be a list of length maux, each element of which is one of:
296     # 'center', 'capacity', 'xleft', or 'yleft' (see documentation).
297
298     amrdata.aux_type = ['center','capacity','yleft']
299
300
301     # Flag using refinement routine flag2refine rather than richardson
            ↪ error
302     amrdata.flag_richardson = False # use Richardson?
303     amrdata.flag2refine = True
304
305     # steps to take on each level L between regriddings of level L+1:
306     amrdata.regrid_interval = 3
307
308     # width of buffer zone around flagged points:
309     # (typically the same as regrid_interval so waves don't escape):
310     amrdata.regrid_buffer_width = 2
311
312     # clustering alg. cutoff for (# flagged pts) / (total # of cells
            ↪ refined)
313     # (closer to 1.0 => more small grids may be needed to cover flagged
            ↪ cells)
314     amrdata.clustering_cutoff = 0.700000
315
316     # print info about each regridding up to this level:
```

```
317     amrdata.verbosity_regrid = 0
318
319     # ----- For developers -----
320     # Toggle debugging print statements:
321     amrdata.dprint = False # print domain flags
322     amrdata.eprint = False # print err est flags
323     amrdata.edebug = False # even more err est flags
324     amrdata.gprint = False # grid bisection/clustering
325     amrdata.nprint = False # proper nesting output
326     amrdata.pprint = False # proj. of tagged points
327     amrdata.rprint = False # print regridding summary
328     amrdata.sprint = False # space/memory output
329     amrdata.tprint = True # time step reporting each level
330     amrdata.uprint = False # update/upbnd reporting
331
332     # More AMR parameters can be set -- see the defaults in pyclaw/data.py
333
334     # ---------------
335     # Regions:
336     # ---------------
337     # rundata.regiondata.regions = []
338     # to specify regions of refinement append lines of the form
339     # [minlevel,maxlevel,t1,t2,x1,x2,y1,y2]
340     rundata.regiondata.regions.append([3, 3, 18000., 100000.,
          ↪ 175.,216.,13.,33.])
```

```
341    # rundata.regiondata.regions.append([3, 3, 8000., 26000.,
           ↪ 270,280,-30,-15])
342
343    # ---------------
344    # Gauges:
345    # ---------------
346    rundata.gaugedata.gauges = []
347    # for gauges append lines of the form [gaugeno, x, y, t1, t2]
348    # rundata.gaugedata.gauges.append([32412, 273.608, -17.975, 0., 1.e10])
349    # rundata.gaugedata.gauges.append([32412, 273.608, -17.975, 0., 100.])
350    rundata.gaugedata.gauges.append([1, 0.2004076800E+03, 0.2189638000E+02,
           ↪ 30000., 100000.]) #keep
351    rundata.gaugedata.gauges.append([2, 0.2021262300E+03, 0.2129147000E+02,
           ↪ 30000., 100000.])
352    rundata.gaugedata.gauges.append([3, 0.2020100500E+03, 0.2172998000E+02,
           ↪ 30000., 100000.]) #keep
353    rundata.gaugedata.gauges.append([4, 0.2030411700E+03, 0.2100330000E+02,
           ↪ 30000., 100000.]) #keep
354    rundata.gaugedata.gauges.append([5, 0.2030075800E+03, 0.2078462000E+02,
           ↪ 30000., 100000.])
355    rundata.gaugedata.gauges.append([6, 0.2033081800E+03, 0.2100178000E+02,
           ↪ 30000., 100000.])
356    rundata.gaugedata.gauges.append([7, 0.2032528300E+03, 0.2086727000E+02,
           ↪ 30000., 100000.])
```

```
357     rundata.gaugedata.gauges.append([8, 0.2033147200E+03, 0.2086845000E+02,
            ↪ 30000., 100000.])
358     rundata.gaugedata.gauges.append([9, 0.2034918800E+03, 0.2061252000E+02,
            ↪ 30000., 100000.])
359     rundata.gaugedata.gauges.append([10, 0.2035076500E+03, 0.2076525000E+02,
            ↪  30000., 100000.])
360     rundata.gaugedata.gauges.append([11, 0.2035282500E+03, 0.2090213000E+02,
            ↪  30000., 100000.]) #keep
361     rundata.gaugedata.gauges.append([12, 0.2040219000E+03, 0.2076123000E+02,
            ↪  30000., 100000.])
362     rundata.gaugedata.gauges.append([13, 0.2049180200E+03, 0.1974170000E+02,
            ↪  30000., 100000.]) #keep
363     rundata.gaugedata.gauges.append([14, 0.2049300300E+03, 0.1974167000E+02,
            ↪  30000., 100000.]) #keep
364     rundata.gaugedata.gauges.append([15, 0.2040984300E+03, 0.2026502000E+02,
            ↪  30000., 100000.])
365     rundata.gaugedata.gauges.append([16, 0.2041583700E+03, 0.2004108000E+02,
            ↪  30000., 100000.])
366     rundata.gaugedata.gauges.append([17, 204.00085, 19.6351, 30000.,
            ↪ 100000.])
367     rundata.gaugedata.gauges.append([18, 203.9681, 19.6678, 30000.,
            ↪ 100000.])
368
369
370     return rundata
```

```
371     # end of function setrun
372     # ----------------------
373
374
375 #-------------------
376 def setgeo(rundata):
377 #-------------------
378     """
379     Set GeoClaw specific runtime parameters.
380     For documentation see ....
381     """
382
383     try:
384         geo_data = rundata.geo_data
385     except:
386         print("*** Error, this rundata has no geo_data attribute")
387         raise AttributeError("Missing geo_data attribute")
388
389     # == Physics ==
390     geo_data.gravity = 9.81
391     geo_data.coordinate_system = 2
392     geo_data.earth_radius = 6367.5e3
393
394     # == Forcing Options
395     geo_data.coriolis_forcing = False
```

```
396
397     # == Algorithm and Initial Conditions ==
398     geo_data.sea_level = 0.0
399     geo_data.dry_tolerance = 1.e-3
400     geo_data.friction_forcing = True
401     geo_data.manning_coefficient =.025
402     geo_data.friction_depth = 1e6
403
404     # Refinement settings
405     refinement_data = rundata.refinement_data
406     refinement_data.variable_dt_refinement_ratios = True
407     refinement_data.wave_tolerance = 1.e-1
408     refinement_data.deep_depth = 1e2
409     refinement_data.max_level_deep = 3
410
411     # == settopo.data values ==
412     topo_data = rundata.topo_data
413     # for topography, append lines of the form
414     # [topotype, minlevel, maxlevel, t1, t2, fname]
415     topo_path = os.path.join(scratch_dir, 'geoclawtopofile.asc')
416     # topo_data.topofiles.append([3, 1, 3, 0., 1.e10, topo_path])
417     topo_data.topofiles.append([3, 1, 3, 0., 150000., topo_path])
418
419     # == setdtopo.data values ==
420     dtopo_data = rundata.dtopo_data
```

```
421     # for moving topography, append lines of the form : (<= 1 allowed for
            ↪ now!)
422     # [topotype, minlevel,maxlevel,fname]
423     #dtopo_path = os.path.join(scratch_dir, 'topofile.tt3')
424     dtopo_path = ('/Users/wiersma/work/geoclaw/'+'dtopo.tt3')
425     dtopo_data.dtopofiles.append([3,3,3,dtopo_path])
426     dtopo_data.dt_max_dtopo = 0.2
427
428
429     # == setqinit.data values ==
430     rundata.qinit_data.qinit_type = 0
431     rundata.qinit_data.qinitfiles = []
432     # for qinit perturbations, append lines of the form: (<= 1 allowed for
            ↪ now!)
433     # [minlev, maxlev, fname]
434
435     # == setfixedgrids.data values ==
436     fixed_grids = rundata.fixed_grid_data
437     # for fixed grids append lines of the form
438     # [t1,t2,noutput,x1,x2,y1,y2,xpoints,ypoints,\
439     # ioutarrivaltimes,ioutsurfacemax]
440
441     return rundata
442     # end of function setgeo
443     # ----------------------
```

```
444
445
446
447 if __name__ == '__main__':
448     # Set up run-time parameters and write all data files.
449     import sys
450     rundata = setrun(*sys.argv[1:])
451     rundata.write()
```

C.3.2 setplot.py

```
1
2  """
3  Set up the plot figures, axes, and items to be done for each frame.
4
5  This module is imported by the plotting routines and then the
6  function setplot is called to set the plot parameters.
7
8  """
9
10 from __future__ import absolute_import
11 from __future__ import print_function
12 import numpy as np
13 import matplotlib.pyplot as plt
14
15 from clawpack.geoclaw import topotools
16 from six.moves import range
17
18 try:
19     TG32412 = np.loadtxt('32412_notide.txt')
20 except:
21     print("*** Could not load DART data file")
22
23 #--------------------------
24 def setplot(plotdata=None):
```

```
25 #--------------------------
26
27     """
28     Specify what is to be plotted at each frame.
29     Input: plotdata, an instance of pyclaw.plotters.data.ClawPlotData.
30     Output: a modified version of plotdata.
31
32     """
33
34
35     from clawpack.visclaw import colormaps, geoplot
36     from numpy import linspace
37
38     if plotdata is None:
39         from clawpack.visclaw.data import ClawPlotData
40         plotdata = ClawPlotData()
41
42
43     plotdata.clearfigures() # clear any old figures,axes,items data
44
45
46     # To plot gauge locations on pcolor or contour plot, use this as
47     # an afteraxis function:
48
49     def addgauges(current_data):
```

```
50        from clawpack.visclaw import gaugetools
51        gaugetools.plot_gauge_locations(current_data.plotdata, \
52             gaugenos='all', format_string='ko', add_labels=True)
53
54
55    #-----------------------------------------
56    # Figure for surface
57    #-----------------------------------------
58    plotfigure = plotdata.new_plotfigure(name='Surface', figno=0)
59
60    # Set up for axes in this figure:
61    plotaxes = plotfigure.new_plotaxes('pcolor')
62    plotaxes.title = 'Surface'
63    plotaxes.scaled = True
64
65    def fixup(current_data):
66        import pylab
67        addgauges(current_data)
68        t = current_data.t
69        t = t / 3600. # hours
70        pylab.title('Surface␣at␣%4.2f␣hours' % t, fontsize=20)
71        pylab.xticks(fontsize=15)
72        pylab.yticks(fontsize=15)
73    plotaxes.afteraxes = fixup
74
```

```
75      # Water
76      plotitem = plotaxes.new_plotitem(plot_type='2d_pcolor')
77      #plotitem.plot_var = geoplot.surface
78      plotitem.plot_var = geoplot.surface_or_depth
79      plotitem.pcolor_cmap = geoplot.tsunami_colormap
80      plotitem.pcolor_cmin = -0.2
81      plotitem.pcolor_cmax = 0.2
82      plotitem.add_colorbar = True
83      plotitem.amr_celledges_show = [0,0,0]
84      plotitem.patchedges_show = 1
85
86      # Land
87      plotitem = plotaxes.new_plotitem(plot_type='2d_pcolor')
88      plotitem.plot_var = geoplot.land
89      plotitem.pcolor_cmap = geoplot.land_colors
90      plotitem.pcolor_cmin = 0.0
91      plotitem.pcolor_cmax = 100.0
92      plotitem.add_colorbar = False
93      plotitem.amr_celledges_show = [1,1,0]
94      plotitem.patchedges_show = 1
95      # for french polynesia
96      plotaxes.xlimits = [181.,241.]
97      plotaxes.ylimits = [-47.,17.]
98      # plotaxes.xlimits = [201.,221.]
99      # plotaxes.ylimits = [-27.,-7.]
```

```
100     # plotaxes.xlimits = [154.,250.]
101     # plotaxes.xlimits = [154.,300.]
102     # plotaxes.ylimits = [-20.,60.]
103     # plotaxes.ylimits = [-40., 53.]
104     # plotaxes.xlimits = [267.267,307.267]
105     # plotaxes.ylimits = [-55.909,-15.909]
106
107     # add contour lines of bathy if desired:
108     plotitem = plotaxes.new_plotitem(plot_type='2d_contour')
109     plotitem.show = False
110     plotitem.plot_var = geoplot.topo
111     plotitem.contour_levels = linspace(-3000,-3000,1)
112     plotitem.amr_contour_colors = ['y'] # color on each level
113     plotitem.kwargs = {'linestyles':'solid','linewidths':2}
114     plotitem.amr_contour_show = [1,0,0]
115     plotitem.celledges_show = 0
116     plotitem.patchedges_show = 0
117
118
119     #-----------------------------------------
120     # Figures for gauges
121     #-----------------------------------------
122     plotfigure = plotdata.new_plotfigure(name='q[0]␣at␣gauges', figno=300, \
123                     type='each_gauge')
124
```

```
125     # Set up for axes in this figure:
126     plotaxes = plotfigure.new_plotaxes()
127     plotaxes.xlimits = 'auto'
128     plotaxes.ylimits = [-1.5, 1.5]
129     plotaxes.title = 'q[0]'
130
131     # Plot q[0] as blue line:
132     plotitem = plotaxes.new_plotitem(plot_type='1d_plot')
133     plotitem.plot_var = 0
134     plotitem.plotstyle = 'b-'
135
136     # Plot topo as green curve:
137     plotitem = plotaxes.new_plotitem(plot_type='1d_plot')
138     plotitem.show = True
139
140     def gaugetopo(current_data):
141         q = current_data.q
142         h = q[0,:]
143         eta = q[3,:]
144         topo = eta - h
145         return topo
146
147     plotitem.plot_var = gaugetopo
148     plotitem.plotstyle = 'g-'
149
```

```
150     def add_zeroline(current_data):
151         from pylab import plot, legend, xticks, floor, axis, xlabel
152         t = current_data.t
153         gaugeno = current_data.gaugeno
154 
155         plot(t, 0*t, 'k')
156         n = int(floor(t.max()/3600.) + 2)
157         xticks([3600*i for i in range(n)], ['%i' % i for i in range(n)])
158         xlabel('Time (hours)')
159         ylabel('Height (cm)')
160 
161     plotaxes.afteraxes = add_zeroline
162 
163 
164 
165     #-----------------------------------------
166 
167     # Parameters used only when creating html and/or latex hardcopy
168     # e.g., via pyclaw.plotters.frametools.printframes:
169 
170     plotdata.printfigs = True # print figures
171     plotdata.print_format = 'png' # file format
172     plotdata.print_framenos = 'all' # list of frames to print
173     plotdata.print_gaugenos = 'all' # list of gauges to print
174     plotdata.print_fignos = 'all' # list of figures to print
```

```
175     plotdata.html = True # create html files of plots?
176     plotdata.html_homelink = '../README.html' # pointer for top of index
177     plotdata.latex = True # create latex file of plots?
178     plotdata.latex_figsperline = 2 # layout of plots
179     plotdata.latex_framesperline = 1 # layout of plots
180     plotdata.latex_makepdf = False # also run pdflatex?
181     plotdata.parallel = True # make multiple frame png's at once
182
183     return plotdata
```

C.4 Run All GeoClaw Models

This Python script runs all files required to implement the GeoClaw model based on the geometries calculated in 6, then save the desired files into a new directory, and then begins the next model, until all five geometries are calculated.

```
1  # -*- coding: utf-8 -*-
2
3  # Import statements
4  import numpy as np
5  import subprocess
6  import glob
7  import shutil
8  import os
9  from variables import moment
10 from maketopo import Maketopo
11 # from linereader import *
12
13 # Absolute path where geoclaw_interface lives
14 path = '/Users/wiersma/work/geoclaw/'
15 subprocess.call(['python', 'geometries.py']) ###Runs the script to make
       ↪ variations in geometry
16 subprocess.call(['python', 'download_topo.py']) ###Runs the script to
       ↪ download bathymetry
17
18 ###names of files containing geometry information
19 file_array=['widegeom.txt', '2widegeom.txt', 'longgeom.txt', '2longgeom.txt'
```

```
    ↪ , 'amorphous.txt']
20
21  ##previously existing .tt3 files cause the program to NOT create a new
    ↪ dtopo file, delete these files
22  for file in path:
23      if file.endswith('.tt3'):
24          os.remove(file)
25
26  for file in file_array:
27      fault_names = np.loadtxt(path+file, usecols=0, dtype='str', delimiter=',
        ↪ ')
28      total=sum(1 for line in fault_names)
29      print total
30      filearea=total*50.e3*100.e3
31      print filearea
32      fileslip=(moment/(4.3e10*filearea))
33      slip = {}
34      for fault in fault_names:
35          print(fault)
36          if fault != 'Name':
37              slip['{}'.format(fault)] = fileslip
38
39      # Name of text file that holds all the earthquake names
40      eq_names = file_array # np.loadtxt("EQNames.txt", dtype='str')
41
```

```
42      # for i in range(len(file_array)): #len(lat)): #len(lat)
43      names=os.path.splitext(file)[0]
44      #get rid of folder to rewrite plots and outputs to
45      if os.path.exists(path+names):
46          shutil.rmtree(path+names)
47
48  # run geoclaw model for input text file
49      mt = Maketopo(slip, names)
50      mt.make_dtopo(True)
51      eq = 'dtopo_{}.tt3'.format(names)
52      subprocess.call(['make', 'topo']) #Call make topo
53      shutil.copyfile(eq, 'dtopo.tt3') # rename the file to generic for
            ↪ setrun
54      subprocess.call(["make", "new"]) # Call make new to clear run
55      #subprocess.call(["make", ".output"]) # Call make .output
56      subprocess.call(["make", ".plots"]) #call make .plots
57
58
59      # AFter Geoclaw runs, copy files to new places before new loop begins
60      path_final = os.path.join(path, "{}".format(names)) # Name of the final
            ↪ path
61      if not os.path.exists(path_final):
62          print('Path␣does␣not␣exist?␣{}'.format(os.path.exists(path_final)))
63          os.mkdir(path_final) # make the new directory
64          #out_folder = os.path.join(path, "_output") # Name of the output
```

```
        ↪ folder
65      # plot_folder = os.path.join(path, "_plots")
66      plot_folder=path+str('_plots')
67      mv_eq = os.path.join(path, "dtopo.tt3") # name of the earthquake
            ↪ file to be moved
68      rename_eq = os.path.join(path_final, '{}.tt3'.format(names))
69      shutil.copy(mv_eq, rename_eq)
70      text_file = os.path.join(path, '{}.txt'.format(names))
71      if os.path.exists(os.path.join(path_final, '{}.txt'.format(names))):
72          text_dest = os.path.join(path_final, '{}_copy.txt'.format(names))
73          shutil.copy(text_file, text_dest)
74      else:
75          shutil.copy(text_file, path_final) # Move the original eq file
                ↪ to the final folder
76
77      #shutil.move(eq, path_final) # Move the original eq file to the
            ↪ final folder
78  if os.path.exists(os.path.join(path_final, '_output')):
79      #out_copy = os.path.join(path_final, '_output_copy{}'.format(
            ↪ eq_names[i]))
80      plot_copy = os.path.join(path_final, '_plots_copy{}'.format(eq_names
            ↪ [i]))
81      #out_folder = os.path.join(path, "_output")
82      plot_out = os.path.join(path, "_plots")
83      #shutil.move(out_folder, out_copy)
```

```
84          shutil.move(plot_out, plot_copy)
85      else:
86          #shutil.move(out_folder, path_final) # Move the _output file to the
                ↪ final folder
87          plot_folder=path+str('_plots')
88          shutil.move(plot_folder, path_final)
89          output_folder=path+str('_output')
90          shutil.move(output_folder, path_final)
```