DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Quantifying the Effect of Initial Conditions on Tsunami Simulation Using Bayesian Methods

Mohamed Houssein Zaghdane

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Quantifying the Effect of Initial Conditions on Tsunami Simulation Using Bayesian Methods

Author:	Mohamed Houssein Zaghdane
Supervisor:	Prof Dr. Michael Bader
Advisor:	Dr. Anne Reinarz
Submission Date:	15/10/2020

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15/10/2020 Mohamed Houssein Zaghdane

Acknowledgments

I wish to express my deepest gratitude to Dr. Anne Reinarz who has constantly helped me tackle on and fix all problems that I encountered during my research.

I would also like to thank my family for the continuous support I have received during the pandemic lockdown.

Abstract

Numerical computations of earthquake generated tsunamis have been carried by multiple researchers in order to recreate and hopefully predict tsunamis. The Okada model is typically used for the tsunami wave generation and the Shallow waters equation model is solved for the wave propagation. The Okada model however requires a lot of uncertain parameters as input. The markov chain Monte-Carlo (MCMC) methods are sample based bayesian methods that can be used to efficiently estimate the probability distributions of unknown parameters in high-dimensionnal problems. In this work we will explore the theory behind tsunami generation and the Metropolis hastings MCMC algorithm and see the effectiveness of this algorithm in inferring initial parameters for Shallow water type problems.

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**

2 Theoretical Background **3**

2.1 The tsunami simulation 3

2.1.1 Wave generation 3

2.1.2 The 2D shallow water equation 3

2.1.3 The Exahype Engine 5

2.2 The Bayesian Inverse problem and Markov Chain Monte-Carlo 7

2.2.1 Bayes Inverse Problem 7

2.2.2 Markov Chains 8

2.2.3 Monte-Carlo methods 8

2.2.4 The Metropolis-Hastings algorithm 9

2.2.5 The MIT Uncertainty Quantification library 10

2.2.6 MultilevelMCMCExaHyPE 10

3 The solitary wave on a beach scenario **12**

3.1 Description 12

3.1.1 Setup 12

3.1.2 Wave parametrisation 13

3.1.3 Experiments 14

3.2 Results and analysis 16

3.2.1 First wave experiments with one unknown parameter 16

3.2.2 The Second wave experiments with two unknown parameters 19

3.2.3 Third wave experiments with three unknown parameters 21

4 The Tohoku Tsunami simulation **25**

4.1 Description 25

4.2 Setup 25

4.3 results 25

5 Conclusion **27**

List of Figures **28**

List of Tables **29**

Bibliography **30**

1 Introduction

Tsunamis are long and powerful waves that are generated by geological events such as landslides, rockslides and earthquakes. Tsunamis can be quite devastating as proven by the Tohoku earthquake tsunami which has caused a death count of 15,641 with 5,007 missing and a lot of structural damage in 2011[1].

Various researchers have worked on numerical computations of earthquake generated tsunamis based on real-world bathymetry. This type of tsunamis is caused by faulting at the bottom of the ocean which causes a displacement at the surface of the water which generates the tsunami wave [2]. For numerical computations of this procedure of wave generation the Okada model is generally used [3]. While different methods for the tsunami propagation exists, the 2D shallow-water equations are appropriate for coastal areas such as Tohoku.

However there exists a lot of uncertainty with these methods. The Okada model requires various parameters such fault strike angle, fault rake angle, fault-dip angle,fault length and fault slip Other factors to consider are the possibly noisy buoy recordings. The Tohoku tsunami that we will be focusing on has a dataset of more than 5300 locations [1], it is nearly impossible to approximate the errors that the recordings may contain. Uncertain initial data also affects the tsunami propagation step as the data used to solve the Shallow-Waters model is directly dependant on the initial data.

Because of the high dimensionality of the unknown parameters space, a numerical approach to solving this problem is inappropriate [4]. A popular approach that has gained popularity these last decades is the Bayes based, Markov chain Monte-Carlo class of algorithms. These methods rely on picking random samples that are used to solve a model, the results returned by the model are then used to pick another sample and the cycle repeats. When ran a high enough amount of time, the candidates picked by the chain can be used to generate probability distributions and draw expectations from our unknown parameter behaviours [5].

In this paper we will present the background behind tsunami simulations and the Markov chain Monte-Carlo methods, in particular the metropolis-Hastings algorithm. We will then run simple simulations with predetermined parameters, try to estimate them using our MCMC algorithm and analyse the results. We will finally run a simulation of the 2011 Tohoku tsunami using real recordings of the waves and the bathymetry.

2 Theoretical Background

2.1 The tsunami simulation

In this section we will discuss the techniques used for numerical computations of tsunamis and the engine we will use to realise our simulations

2.1.1 Wave generation

Tsunamis are caused by fault deformations at the bottom of the ocean which causes a water surface displacement. Tanioka and Satake have documented the effects of the fault and the horizontal movement of the slope on the amplitude of the generated tsunami waves [2]. The surface displacement in an elastic half-space can be calculated using the Steketee equation

$$u_k = \frac{1}{F} \iint\limits_{\Sigma} \Delta u_i [\lambda \delta_{ij} u_k^{n,n} + \mu (u_k^{i,j} + u_k^{j,i})] v_j d \sum \tag{2.1}$$

where λ and μ are lame's constants, δ_{ij} is Kronecker's delta v_j is the direction cosine of the normal to the fault surface element $d \sum$. u_k^i is the kth component of the surface displacement caused by the ith point force which is of magnitude F [2].

The Okada model, which is based on this equation, is generally used in tsunami simulations in order to generate the initial wave. When supplied with multiple parameters such as the fault's rectangular geometry namely the fault's length, dip, width, strike and depth. The okada model calculates the initial sea surface displacement in an elastic half-space [3].

2.1.2 The 2D shallow water equation

For the propagation stage of the tsunami wave, we will use the shallow water equations as they are appropriate for tsunami scenarios [7]. The 2-dimensional shallow water model (SWE) is a system of hyperbolic partial differential equations that model the flow of flat water. The equations neglect vertical velocities which requires the vertical scale of the simulated domain to be a lot smaller than the horizontal scale for the simulation to be accurate. This is particularly useful for simulating coastal regions [8].

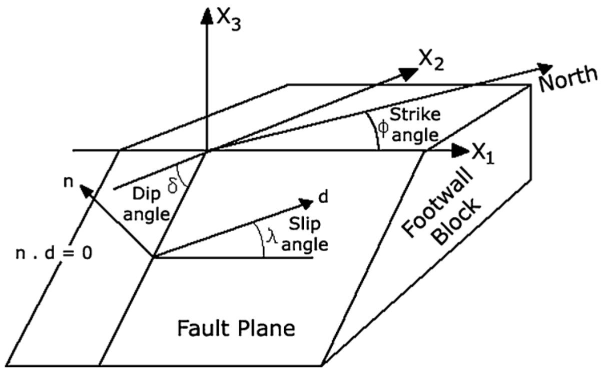

Figure 2.1: A visualisation of the fault parameters used in the okada model, image taken from [6]

The SWE are derived from the laws of conservation of mass and linear momentum. The Exahype engine, which we will be using for the tsunami simulation, offers the following formulation of the Shallow water equations :

$$\frac{\partial}{\partial t}\vec{Q} + \frac{\partial}{\partial x}\vec{F_1}\left(\vec{Q}\right) + \frac{\partial}{\partial y}\vec{F_2}\left(\vec{Q}\right) + \vec{B}(Q).\nabla Q = 0 \tag{2.2}$$

$$\vec{Q} = \begin{pmatrix} h \\ hu \\ hv \\ b \end{pmatrix}, \vec{F_1} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \\ 0 \end{pmatrix}, \vec{F_2} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \\ 0 \end{pmatrix}, \vec{B}.\nabla Q = \begin{pmatrix} 0 \\ hg\partial_x b + \\ hg\partial_y b \\ 0 \end{pmatrix} \tag{2.3}$$

where :

- $\vec{Q}$ are our conserved quantities, $\vec{F_1}$ and $\vec{F_2}$ are the fluxes, and $\vec{B}(Q).\nabla Q$ is the non-conservative part.
  - h is the height of the water column.
  - u and v represent the horizontal velocities respectively on the x-axis and the y-axis
  - b is the bathymetry. The bathymetry refers to the depth of the seafloor relative to the surface of the ocean. It is used to communicate data about the underwater terrain [9].
  - g is the gravity. A constant value of $9.81m/s^2$ is typically used.

2.1.3 The Exahype Engine

The exahype engine (Exascale Hyperbolic PDE engine) is a high performance engine for solving first order hyperbolical partial differential equations. The engine gives access to multiple numerical solvers and an intuitive way to implement and simulate hyperbolic PDE systems which may contain conservative and non-conservative terms as long as they are given in this form :

$$\frac{\partial}{\partial t}Q + \nabla.F(Q, \nabla Q) + B(Q).\nabla Q = S(Q) + \sum_{i=1}^{n_{ps}} \delta_i \tag{2.4}$$

where Q is the state vector containing our conserved variables, $F(Q)$ is the flux tensor and $B(Q)$ represents is the non conservative part of the equation, $S(Q)$ is the source term and δ_i are the point sources [8].
Exahype already includes implementations of the shallow water equations as mentioned in the section before 2.2 which we will be using for our wave simulations. As seen when comparing both equations 2.2 and 2.4, The right part of the form is trivial for the SWE and is equal to zero. The steps to implementing and running a simulation are the following

- Setting up the configuration. While it contains many options, the most relevant part is setting our physical and temporal domain, the solvers used and our plotters, which allows us to generate Vtk files to visualise our simulations using tools such as Paraview but also to record cell data at certain points for successive time-steps which can be used to simulate buoy recordings.
- Setting up the initial values for our quantity of interest Q. For simple simulations is it possible to set these values up manually by setting up Q for each cell. For advanced simulations that recreate real environments with complex geography we use ASAGI, which is a library that allows us to easily query and set up complex geoinformation [10].
- Setting up appropriate boundary conditions which define the behavior of boundary cells.
- Setting up the eigenvalues, fluxes and non conservative products in the predefined appropriate functions in accord with the shallow water equations, an example of the intuitive syntax used is presented in code snippet.
- finally we compile and run the simulation. Exahype offers parallelization and scaling abilities using Intel's MPI and Threading Building Blocks (TBB). This means that we can highly accelerate the simulation by using multiple nodes and cores. We will use the Lrz linux clusters [11] to take advantage of the Exahype scaling features. We can then use the output Vtk files to visualise the simulations.

```
f[0] = vars.hu();
f[1] = vars.hu() * vars.hu() * ih + 0.5 * grav * vars.h() * vars.h();
f[2] = vars.hu() * vars.hv() * ih;
f[3] = 0.0;
g[0] = vars.hv();
g[1] = vars.hu() * vars.hv() * ih;
g[2] = vars.hv() * vars.hv() * ih + 0.5 * grav * vars.h() * vars.h();
g[3] = 0.0;
```

Listing 2.1: A snippet of the implementation of the flux function in exahype, the array f represents $\vec{F_1}$ and g represents . vars is a variable that contains all our quantities of interests Q

2.2 The Bayesian Inverse problem and Markov Chain Monte-Carlo

In this section we will explain the Bayesian inverse problem and How the Markov Chain Monte-Carlo algorithms are used to approach this type of problem

2.2.1 Bayes Inverse Problem

For our simulations, we wish to quantify unknown parameters that affect the outcome of a tsunami simulation. We can write the equation we want to solve as

$$y = O \circ G(\theta) + \eta \tag{2.5}$$

with $\theta \; in \; \mathbb{R}^n$ and $\eta, y \; in \; \mathbb{R}^m$

- O is the observation operator : Buoys that record water height
- θ a set of parameters which are unknown in our situation
- G is the model : the tsunami
- η measurement errors
- y is the observed data : The output of our buoys

We possess the data y and we wish to infer the unknown parameters but solving this equation is impossible as the problem is ill-posed [12] :

- We do not know The extent of the measurement error η and cannot approximate it which means that we can not subtract it from the observed data y to be able to invert accurately the model G. This makes deriving accurate unknown parameters θ impossible because of the uncertainty caused by the error.
- In case the dimension of the unknown parameters is higher than that of the solution $n >> m$, the system is underdetermined, which means we can have multiple solutions [13].

To solve this problem we will take an approach based on the Bayes theorem, which is often presented as follows :

$$f(\theta | O \circ G(\theta) + \eta = y) = \frac{f(y|\theta).f(\theta)}{f(y)}. \tag{2.6}$$

- $f(\theta)$ is the prior which represents our prior knowledge of θ
- $f(y|\theta)$ is the likelihood which represents the probability of observing the data y given a model M
- $f(\theta|O \circ G(\theta) + \eta = y)$ is the posterior which is what we learned about θ after observing y
- f(y) is the evidence which serves as a normalising constant [14]

While there are multiple approaches based on Bayesian statistics, in our situation, we need an optimised algorithm that can work with high-dimensional parameter space and with no prior knowledge about our unknown. The Markov Chain Monte-Carlo family of alghorithms represent a good option in this case. [4]

2.2.2 Markov Chains

A Markov Chain is a series of random variables X^1,X^2,...X^n Where the value of $X^k + 1$ is only influenced by X^k. This can be written as :

$$P(X^{n+1}|X^0, X^1, ...X^n) = P(X^{n+1}|X^n) \tag{2.7}$$

For that we only need the marginal distribution for x^0, the probabilities of each following possible state and the transition probability to move from one state X^k to another $X^k + 1$. Eventually a Markov Chain can reach a stationary distribution π which is unchanged by transitions. This can be formally written as

$$\pi(x) = \sum_{\tilde{x}} \pi(\tilde{x}) T_n(\tilde{x}, x) \tag{2.8}$$

for a state x and all possible transitions n. All finite Markov Chains possess at least one stationary distribution.
This stationary state is ideal to estimate expectations for our sampled parameters [4].

2.2.3 Monte-Carlo methods

Monte-Carlo methods are computational algorithms that rely on repeated random sampling from probability distributions. The Experiment must be repeated a high amount of time in order to reach our desired quantity of interest. These methods are particularly efficient when working with complex models and are parallelizable which decreases the execution time [15]. Monte-Carlo methods typically follow these 3 steps [16]:

1. n Independant samples of the random parameters are generated from a certain probability distribution and within their defined domains

2. For each sample, the model G is solved and the result recorded
3. After all samples have been run, the expectation of the random parameters is calculated.

2.2.4 The Metropolis-Hastings algorithm

Markov Chain Monte-Carlo (MCMC) is a class of algorithms based on a combination of Monte-Carlo methods and Markov Chains. A Markov Chain Monte-Carlo is thus a sampling algorithm where $\forall$ k $\in$ [0 .. (n-1)] the selection of the next candidate set of parameters $X^k + 1$ depends solely on X^k, which means that each sample drawn becomes dependant of the previous sample.
Markov Chain Monte-Carlo methods are playing a big role in different domains such as machine learning and statistics [5], as they are typically more efficient than numerical methods for problems with a high dimensionality [4]. The MCMC method that we use is the classic Random-walk Metropolis-Hastings algorithm (MH). This method uses rejection samplings which means that for each candidate sample, we will either accept or reject this candidate and re-sample depending on the likelihoods of the actual and candidate samples. This helps make sure that the sampler tends to visit high probability samples more frequently than low probability samples, while also not get stuck in certain value spaces [5]. As initialisation for this implementation, we need to first pick an initial sample set of parameters X^0 = [θ_0, θ_1, .. , θ_n] and a proposal distribution.

We will use a Gaussian distribution Normal(0, *sigma*) as our proposal distribution where *sigma* represents the covariance. The Gaussian distribution is a symmetric distribution which is why our MH algorithm is called a "random walk Metropolis-Hastings". A suitable covariance value must be selected. If the value if too small, the acceptance rate will be too high and the chain will move slowly and not cover all the domain. On the contrary if its value is too high, the acceptance rate will be too low and the chain will not move. A perfect value must be found to reach a medium acceptance rate which is achievable through trial and error [17].

The Metropolis-Hastings algorithm is a popular Markov Chain Monte-Carlo method typically follows the following steps :

1. a new candidate sample X^1 = [θ'_0, θ'_1, .. , θ'_n] is considered. We calculate the acceptance ratio α. α is a probability based on the likelihood of the new distribution X^1, the likelihood of the old distribution X^0, and the proposal distributions $Q(X^0 \mid X^1)$ and $Q(X^1 \mid X^0)$ (2.9).
2. We then generate a random uniform number r in [0,1].
   - If r < α, the new candidate is rejected and another one is chosen.
   - If r > α, the candidate is accepted and X^1 will be used for sampling the next candidate.

3. If we have no reached our desired total number of samples, we repeat from step 2.

$$\alpha = \min\left(1, \frac{\pi(X^1|y)Q(X^0|X^1)}{\pi(X^0|y)Q(X^1|X^0)}\right) \tag{2.9}$$

The Acceptance function has this form to guarantee that the stationary distribution obtained is the one we are interested in [18].

When a high number of samples are run, the Markov Chain will approach a stationary distribution π. The chain of candidates picked by the sampler can be used to approximate the estimated values of our parameters.

Because the initial set of values is random and the prior distribution is uninformed, the Markov Chain needs a certain amount of samples before reaching realistic candidates. The initial values may be extremely unrealistic and affect our parameter probability distribution, which is why we discard a part from the start of the chain. This is typically called the "burn-in" [5].

After running the Algorithm with N Samples, we wish to obtain the expectation for our unknown quantities, the Markov Chain strong law ensures that [19]:

$$E_{p(x|y)}(f(x)) = \int f(x)p(x|y)dx \tag{2.10}$$

we can use this equation to calculate the mean of our parameters by using $f(x) = x$ [5].

$$\mu_n = \frac{1}{N}\sum_{i=1}^{n} f(x_i) where f(x_i) = x_i \tag{2.11}$$

2.2.5 The MIT Uncertainty Quantification library

MUQ (MIT Uncertainty Quantification) [20] is a library for uncertainty quantification. It offers multiple advanced Markov Chain Monte-Carlo algorithms for computationally expensive models through parallel approximations.

MUQ offers multiple implementations of MCMC methods, which of course include our used Random-walk Metropolis-Hastings algorithm. The library offers tools to generate a proposal distribution and configure our MCMC with the appropriate settings (Sample number, burn in ..). Values of interest are then returned such as the mean and the Quantity of interest.

2.2.6 MultilevelMCMCExaHyPE

MultilevelMCMCExaHyPE [21] is a MUQ interface for Exahype. It allows us to configure our MCMC settings and pass the configuration directly to MUQ and serves as communicator that allows Exahype to solve the model during The MCMC steps. After The MCMC process running on MUQ generates a new candidate sample, the parameters are passed to the MultilevelMCMCExaHyPE which we can use to discard problematic samples and pass acceptable parameters to Exahype where they are used to alter the initial data. A simulation is then run and data collected by probes at relevant coordinates is then returned and used to calculate the likelihood of the sample. This likelihood is then used by MUQ for the acceptance function.

Log files are generated by the software which contain the parameters and likelihoods of each sample that was generated. We can use these files to generate relevant plots for our analysis.

3 The solitary wave on a beach scenario

3.1 Description

Our first simulation will be a simple wave that crashes on a beach. The wave starts at a relatively close distance from the shoreline which makes the shallow water equations appropriate as a model to calculate the wave's propagation. This scenario has simple and easily modifiable initial parameters, which means we can generate data from a reference simulation and then compare it to the output of the MCMC .We will use this scenario for multiple experiments to test the effectiveness of our MCMC implementation.

3.1.1 Setup

For this simulation, we will use the Exahype engine with an ADER-DG solver [8] and the Shallow water equations as our model. Exahype already includes examples for the SWE which have implementation for the flux, eigenvalues and non-conservative products. We will use wall boundary conditions which means that waves are reflected on the boundaries of the simulation. We only need to adapt the initial data and configuration file to our needs.

As the scenario is simple, we will not need to use a parser for the data. The conserved quantities Q for each cell will be set up manually using a combination of parameters. All relevant parameters are documented in 3.1, The unknown parameters that we will infer with the Markov Chain Monte-Carlo will be picked from them. The velocity on the y-axis is set to zero for all cells at all time t for this simulation.

The configuration file lets us set up most of the settings for our simulations, the most relevant settings are :

- The dimensional domains : $X \in [-10, 60]$, $Y \in [0, 1]$ and $T \in [0, 0.1]$ which respectively represent the horizontal axis, the vertical axis and time. The low time domain is chosen in order to highly speed up each simulation and doesn't affect our MCMC.
- Plotters : The locations of the plotters will be adapted depending on the simulations and they will serve a similar role as buoys which record the wave height at all times
- The mesh size: This setting allows us to highly decrease the time-cost of each simulation but can lead to very rough output. We will use a relatively high mesh size in order to

accelerate our MCMC

After finishing the configuration, We first run our reference wave simulation with Exahype only. This simulation's initial data will be set with the real parameters and we will use it to get multiple probe recordings that we can use later as a reference for our MCMC's likelihood function.

The second step is to configure our Markov chain Monte-Carlo. We will use the MultilevelMCMC software we mentioned in the previous chapters. We set up our desired number of samples and burn-in which we will vary depending on the experiment. We then set the start values of our unknown parameters to a random number within their domain, we choose values that do not reflect the real values of the parameters. We finally set an appropriate likelihood function which is as follow :

$$l = e^{\sum_0^n -\frac{1}{n} \times 0.5 \times d^2 \times 500} \tag{3.1}$$

- d is the average difference between our reference probe recordings and the probe recordings output using our sampled set of parameters.
- n is the number of used probes which we will vary depending on the experiment.

As mentioned in previous chapters, our MCMC algorithm will be the Metropolis–Hastings algorithm with a Gaussian proposal. The initial values for our unknown parameters will be random. After MUQ select candidate parameter samples, we will discard negative values selected and not set any upper bound for the selected value. Values inferior to 0.1 for water depth d are also discarded as values close to zero cause the Exahype simulation time to be highly increased.

Following the end of the execution of our MCMC, we will use self-written code based on Plotting related python libraries (Matplotlib, Numpy, Pandas, Seaborn) to analyse the expectation for each parameter. Most relevant plots are the parameter density plots and the MCMC iterations.

3.1.2 Wave parametrisation

We will try our MCMC with three different sample dimensions. The wave has a multitude of initial values that can be used as parameters, their names ,what they represent and their correlations are represented in the following graph.

Parameter name	Description	affected by
d	The water depth on undisturbed sea	$\varnothing$
h	The wave's initial peak height (affected by the water depth d)	d
β	The angle of the ground elevation along the coastline	$\varnothing$
Γ	Affects the initial shape of the wave	h, d
X0	Length of the ground elevation along the coastline	d, β
L	Distance of the wave from the beach	Γ, d
eta	The water viscosity	h, L, d, Γ

Table 3.1: All parameters that affect our wave's initial shape and how they are correlated to each others

These parameters help set up the initial values for each cell's conserved data Q. We will use different number of parameters from the table. Depending on the parameters chosen we will place our probes at appropriate locations that allow for meaningful output to infer our parameter (the beta parameter cannot be inferred with no probe near the coastline).

The experiments will be compared to the same reference simulation, where the wave is initially located at $x3$. We will do three base experiments with increasing unknown parameters to see how the increasing dimensionality of the set of parameters affects results of the MCMC.

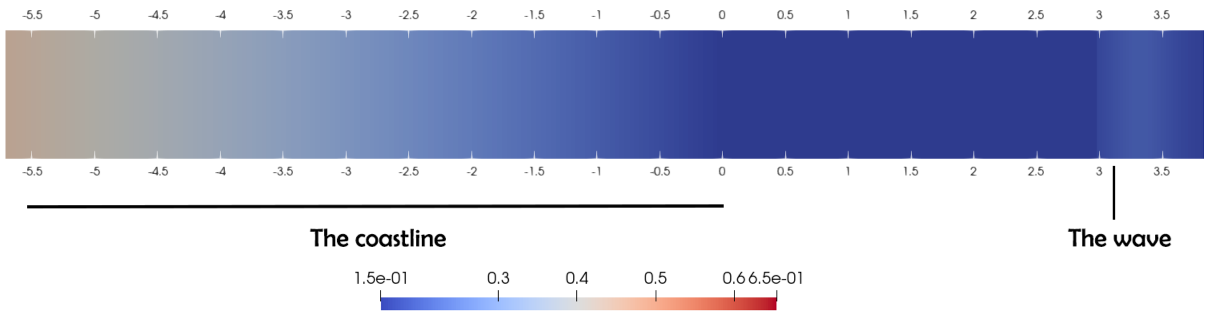

Figure 3.1: Rendering of the wave at initial state using Paraview, coloring depends on the total height of the water and the bathymetry

3.1.3 Experiments

First wave experiments with one unknown Parameter

For the first wave we will be only using a single parameter, which directly affects the maximum water height h. We will use two probes at respective 2D coordinates $\begin{pmatrix}3.1\\0.4\end{pmatrix}$ and $\begin{pmatrix}3.3\\0.5\end{pmatrix}$.

These coordinates are chosen specially to output relevant data on the wave.

We will then run a MCMC experiment with four probes, respectively at $\begin{pmatrix}2.0\\0.5\end{pmatrix}$, $\begin{pmatrix}3.0\\0.5\end{pmatrix}$, $\begin{pmatrix}3.2\\0.5\end{pmatrix}$ and $\begin{pmatrix}3.4\\0.5\end{pmatrix}$. We want to use this experiment to study the effect irrelevant probe data, which means that it's output unaffected by our unknown parameter h, on the speed at which our Markov Chain reaches a stationary distribution as irrelevant probe data directly affects our likelihood function.

Parameter name	Real value
h	0.3

Table 3.2: The first wave's unknown parameter and its real value

Second wave experiments with two unknown Parameter

For the second wave we will be trying to infer two parameters, The water depth d and wave height h. The goal for this choice of parameters is to see the effect of choosing two correlated parameters as h's value is directly affected by d. We will use probes at respective coordinates $\begin{pmatrix}2.0\\0.5\end{pmatrix}$,$\begin{pmatrix}3.0\\0.5\end{pmatrix}$,$\begin{pmatrix}3.2\\0.5\end{pmatrix}$,$\begin{pmatrix}3.4\\0.5\end{pmatrix}$ and $\begin{pmatrix}3.6\\0.5\end{pmatrix}$. All probes deliver relevant data.

Parameter name	Real value
d	0.15
h	0.3

Table 3.3: The second wave's experiments parameters

First wave - Three Parameters

Finally we use three parameters : the water height h, the water depth d and Γ which affects our initial wave's shape. as seen in the table 3.1. The goal of this experiment to see how the addition of a third highly correlated parameter affects the chain.

Parameter name	Real value
d	0.15
h	0.3
Γ	3

Table 3.4: The third wave's experiments parameters

3.2 Results and analysis

3.2.1 First wave experiments with one unknown parameter

For our first MCMC run, we have only used two probes. These probes have been selected to output relevant data about the wave. For only 1000 samples, our mean and posterior distribution have converged to a mean close to our real value. The density plot 3.4 shows a high density at values close to our parameter value. A higher burn-in value for this experiment shows a mean closer to the real value, which is documented in the table below.

Number of samples	1000	1000
burn in	100	300
Parameter	h	h
Real value	0.3	0.3
output mean value	≈ 0.3512	≈ 0.33
Error	$\approx 0{,}0519$	≈ 0.03

Table 3.5: First wave results with use of only two probes

For our second experiment Which uses a higher number of probes, the chain clearly takes a longer time to reach an acceptable distribution of our value expectation. The irrelevant probe data added affects our likelihood function which causes normally unlikely sample values to be more frequently visited. Increasing the number of samples and the burn-in causes our expectation and mean to look more likely and similar to the first experiment.

If we study the MCMC iteration graph 3.7 we can see that the iterations converge towards the real value of our parameter after approximately 8000 samples. Setting the burn-in to 9000 shows that the mean value is nearly perfectly equal to our real parameter and the graph 3.6 shows high density around this value.

Number of samples	1000	12000	12000	12000
burn in	100	1000	5000	9000
Parameter	h	h	h	h
Real value	0.3	0.3	0.3	0.3
output mean value	≈ 0.5	≈ 0.4	≈ 0.3546	≈ 0.3054
Error	≈ 0.2	≈ 0.1	≈ 0.0546	≈ 0.0054

Table 3.6: First wave results with use of four probes

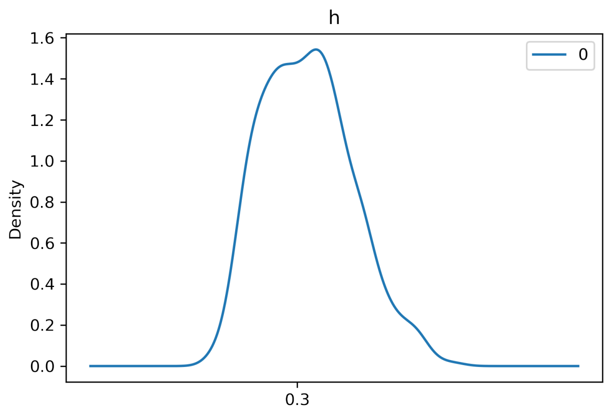

Figure 3.2: 1st wave experiment: Density plot of the h parameter in our first experiment for 2 probes and for a burn-in = 100

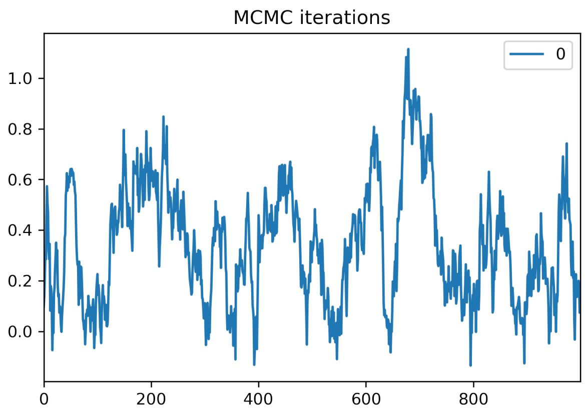

Figure 3.3: 1st wave experiment: Sample iteration of the MCMC for the h parameter and 2 probes

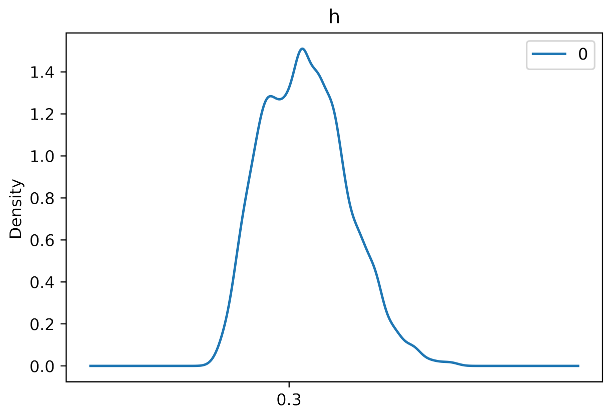

Figure 3.4: 1st wave experiment: Density plot of the h parameter in our first experiment for 2 probes

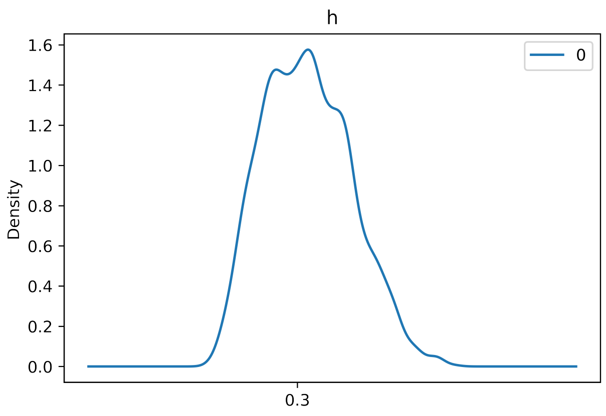

Figure 3.5: 1st wave experiment: Sample iteration of the MCMC for the h parameter and 2 probes

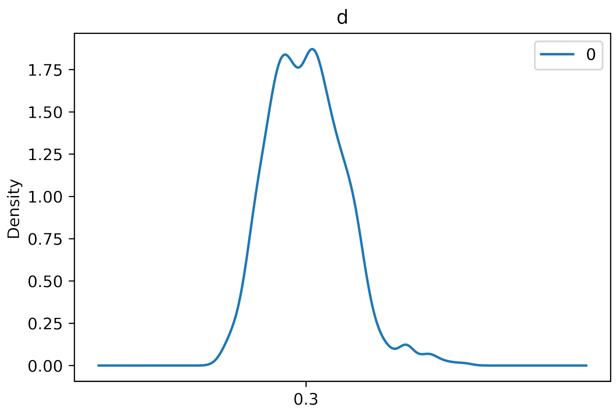

Figure 3.6: 1st wave experiment: Sample iteration of the MCMC for the h parameter and 4 probes

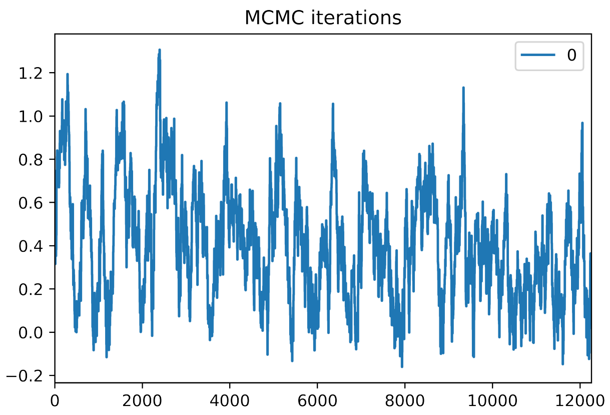

Figure 3.7: 1st wave experiment: Sample iteration of the MCMC for the h parameter and 4 probes

3.2.2 The Second wave experiments with two unknown parameters

For this experiment we want to infer two parameters, h and d. When we analyse the data collected in the table 3.7, we can make the following observations :

- The burn-in value barely has any effect on the means.
- the mean of the h value is far from the real value of our parameter, However an observation of the figure 3.10 shows that their are two zones with a relatively high density with second one being approximately around the value of 1.3. This means that the mean is not representative of the actual highest probability value which approaches more the value of 0.19
- The d parameter converges faster to a credible interval at around 0.2. By looking at both iteration plots 3.9 and 3.11 we see that h gets away from values close to the real parameter and that a lot of samples are generated around a far away value. If we consider that d is an independent parameter and h's value is directly affected by d, we can deduce that this correlation causes deviation for the chain.

The 2D density plot **??** displays clearly that the highest probability zones are around our real parameter values. Despite a bigger margin of error, the chain results are acceptable for this experiment.

Number of samples	15000	15000
burn in	1000	5000
Parameter	d	d
Real value	0.15	0.15
output mean value	≈ 0.2132	≈ 0.21
Error	≈ 0.0632	≈ 0.06
Parameter	h	h
Real value	0.3	0.3
output mean value	≈ 0.51	≈ 0.54
Error	≈ 0.21	≈ 0.23

Table 3.7: Second wave experiment results

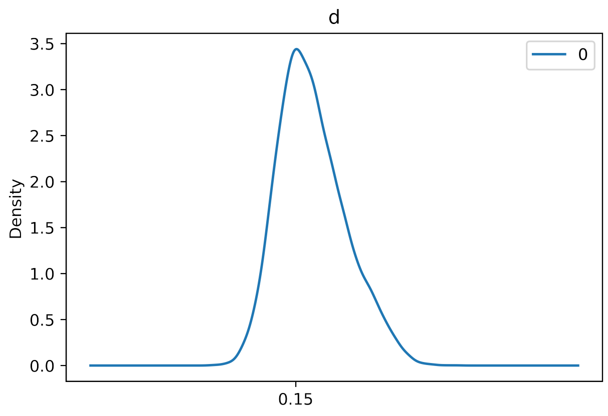

Figure 3.8: 2nd wave experiment: Density of the d parameter for a burn-in=5000

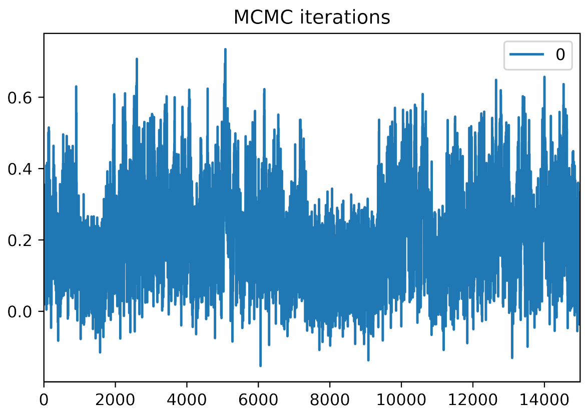

Figure 3.9: 2nd wave experiment: Sample iteration for the d parameter

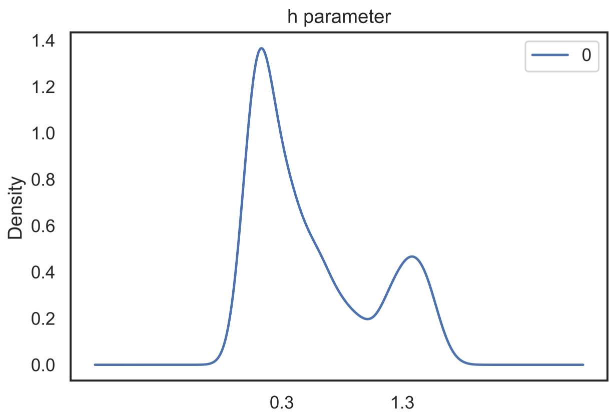

Figure 3.10: 2nd wave experiment: Density of the h parameter for a burn-in=5000

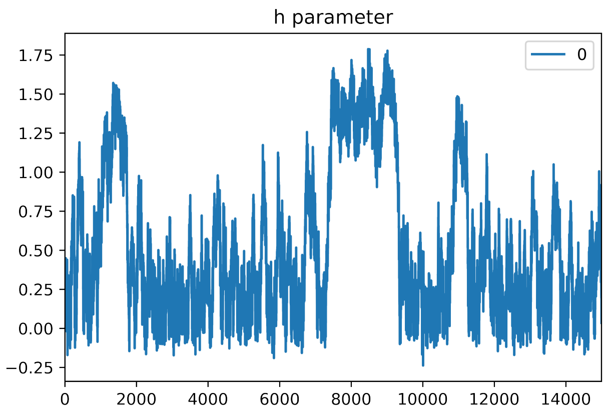

Figure 3.11: 2nd wave experiment: Sample iteration for the h parameter

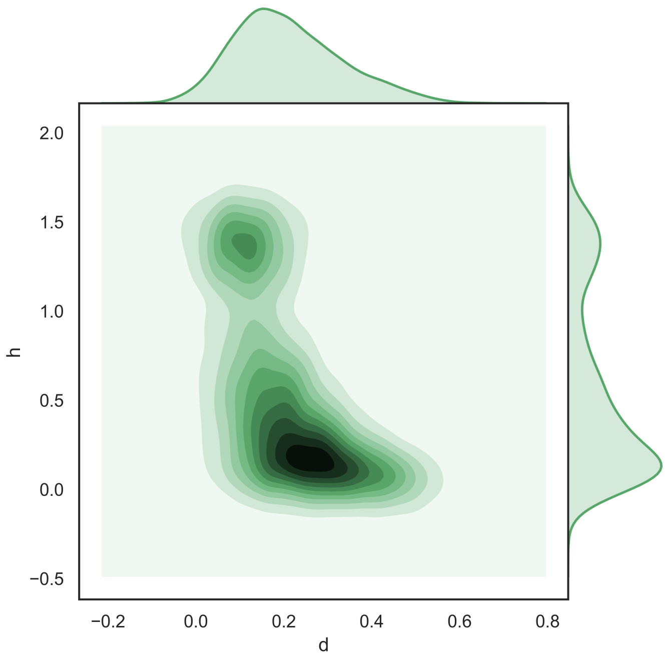

Figure 3.12: 2nd wave experiment: 2D density of the unknown parameter d and h for a burn-in=5000

3.2.3 Third wave experiments with three unknown parameters

For the third experiment we use three highly correlated parameters. This experiment is the most inconclusive of the three experiments. When we analyse the density plot of the *d* parameter we see a highly out of the mark high probability interval which is around the value of 3. The Γ and h parameters however display realistic density plots. Drawing conclusions regarding the reasons of these results is hard as multiple attempts of running the MCMC have led to similarly unsatisfactory results. The following reasons can be considered in our situation :

- The MCMC may need an extremely high number of samples to stabilize because of the high correlation of the parameters. This was sadly impossible to prove as the MCMC runs take an extremely long time to finish executing and we couldn't at the time of writing this thesis reach a high enough sample number.
- Our MCMC likelihood output has shown multiple similar likelihood values for most of the samples. While this may clearly seem as an error, the MCMC software relies on Intel's Mpi to run multiple communicating processes to highly accelerate the execution. This has shown problems as some 'NaN' and unexpected values have appeared during some of the experiments but the errors caused are inconsistent and should also affect our other experiments.
- the samples generated may affect Exahype's output. To maximise the generated number of samples for our experiments, we have made sure to make the mesh size as rough as possible without influencing the quality of the probe output. The different sample values may have an effect on the quality of the output.

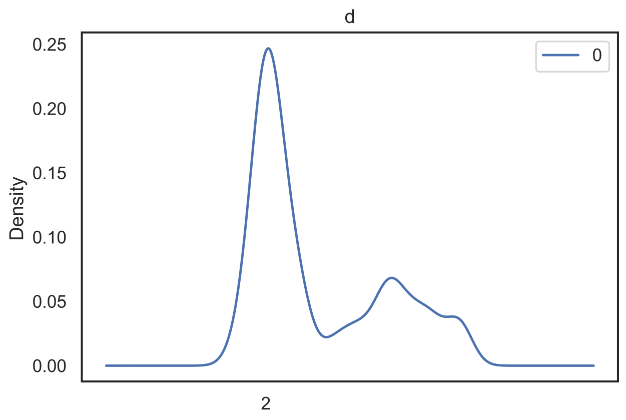

Figure 3.13: 3rd wave experiment: Density of the d parameter for a burn-in=5000

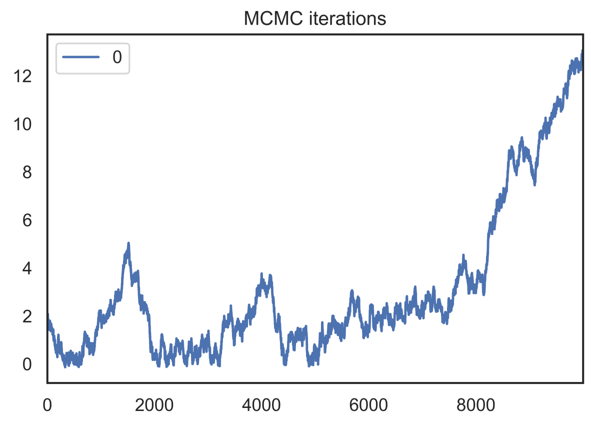

Figure 3.14: 3rd wave experiment: Sample iteration for the d parameter

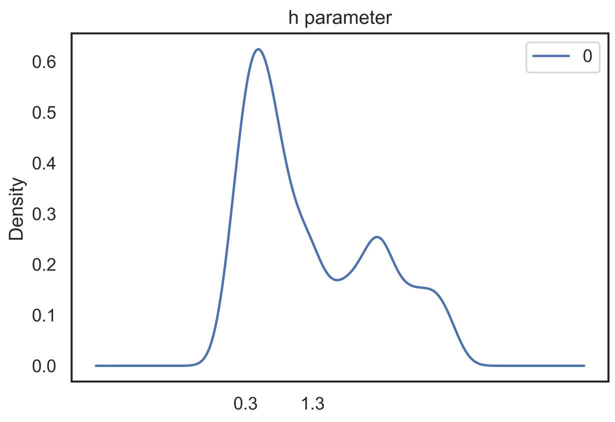

Figure 3.15: 3rd wave experiment: Density of the h parameter for a burn-in=5000

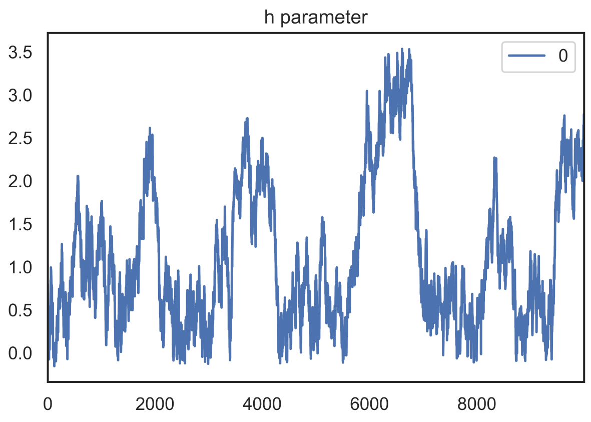

Figure 3.16: 3rd wave experiment: Sample iteration for the h parameter

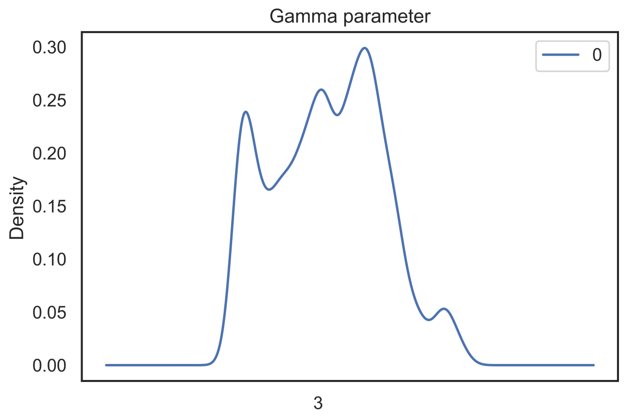

Figure 3.17: 3rd wave experiment: Density of the Γ parameter for a burn-in=5000

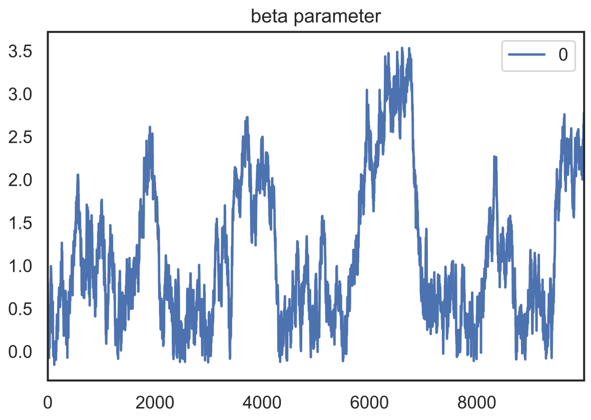

Figure 3.18: 3rd wave experiment: Sample iteration for the Γ parameter

4 The Tohoku Tsunami simulation

4.1 Description

In this Scenario we will simulate the tohoku Tsunami that hit the pacific coasts of japan in 2011 [1]. This simulation is based on the bathymetry of the Tohoku coast and real data recorded by multiple boys along the ocean during the catastrophy.

4.2 Setup

For this simulation we will use Exahype. As this is a complex scenario, we will have to use extra libraries to input our initial data. This requires the usage of ASAGI as mentioned in previous chapters with other software such as Yaml-cpp, Easi and ImpalaJIT. Similarly to the wave equation, we will use an ADERDG solver and the shallow-water equations as our model.
Next we set the configuration file up. In contrast to the simple wave simulation we need high dimensions which we will initially set at $X \in [-500, 1000]$, $Y \in [-750, 750]$ and $T \in [0, 10]$. We will use a relatively fine mesh size to increase the clarity of our output when visualised.

4.3 results

At the moment of writing this thesis, this experiment has unfortunately been inconclusive. An unexpected error which causes the simulation to generate 'NaN' values at t>0 could not be resolved as shown in figures 4.1 and 4.2. Different dimensional domains cause different types of errors. Because of the usage of multiple dependencies in order to parse the data needed for the simulation, it was not possible to pinpoint what seems to be causing the error.

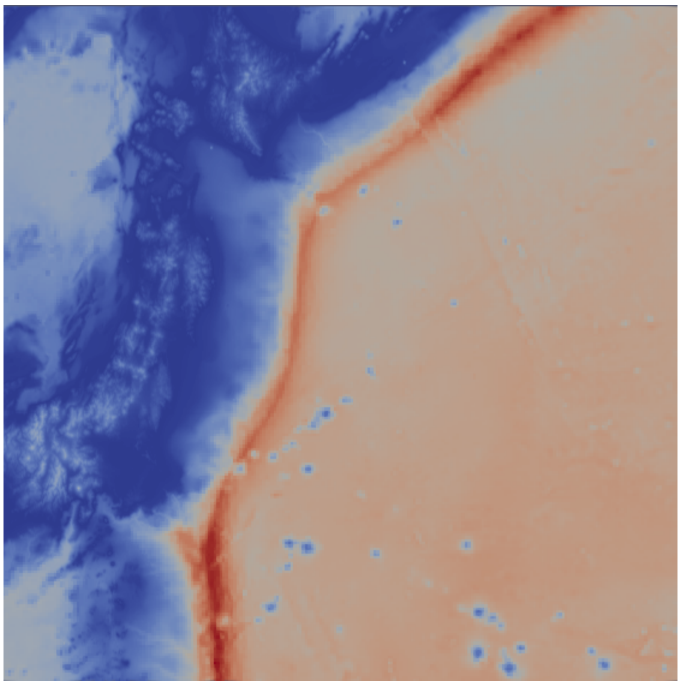

Figure 4.1: The initial state of our Tohoku simulation at t=0

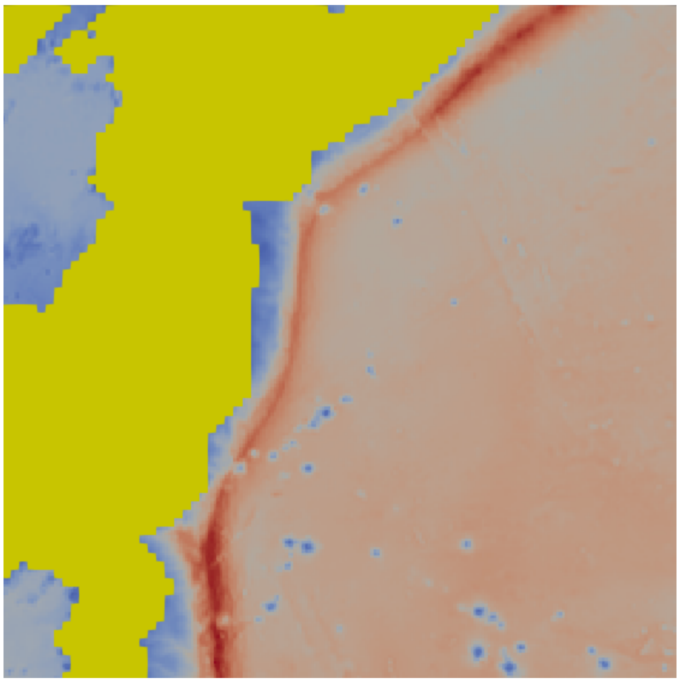

Figure 4.2: The Tohoku simulation at any time t>0, the simulation presents unexpected NaN values

5 Conclusion

The aim of this thesis was to see how effective Markov Chain Monte-Carlo methods in quantifying unknown parameters for tsunami simulations. For our first experiments, The Metropolis-Hastings algorithm has shown quite satisfactory results with approximating unknown parameters in a relatively low number of samples despite no prior knowledge of the tendencies of the parameters we were looking for. The third experiment however displayed odd results and could not be used to estimate to effectiveness of the MCMC algorithm because of too much uncertainty. The tsunami simulation has unfortunately displayed errors that prevented further experimentation with quantifying high-dimensional unknown parameter space.

For future work on this subject, I would of course recommend continuing where this thesis left off and that is the Tsunami simulation. I would also recommend finding an appropriate likelihood function that can discard irrelevant probe data while still accurately calculating the sample likelihood.

List of Figures

2.1 A visualisation of the fault parameters used in the okada model, image taken from [6] 4

3.1 Rendering of the wave at initial state using Paraview, coloring depends on the total height of the water and the bathymetry 14
3.2 1st wave experiment: Density plot of the h parameter in our first experiment for 2 probes and for a burn-in = 100 17
3.3 1st wave experiment: Sample iteration of the MCMC for the h parameter and 2 probes 17
3.4 1st wave experiment: Density plot of the h parameter in our first experiment for 2 probes 17
3.5 1st wave experiment: Sample iteration of the MCMC for the h parameter and 2 probes 18
3.6 1st wave experiment: Sample iteration of the MCMC for the h parameter and 4 probes 18
3.7 1st wave experiment: Sample iteration of the MCMC for the h parameter and 4 probes 18
3.8 2nd wave experiment: Density of the d parameter for a burn-in=5000 20
3.9 2nd wave experiment: Sample iteration for the d parameter 20
3.10 2nd wave experiment: Density of the h parameter for a burn-in=5000 20
3.11 2nd wave experiment: Sample iteration for the h parameter 21
3.12 2nd wave experiment: 2D density of the unknown parameter d and h for a burn-in=5000 21
3.13 3rd wave experiment: Density of the d parameter for a burn-in=5000 22
3.14 3rd wave experiment: Sample iteration for the d parameter 23
3.15 3rd wave experiment: Density of the h parameter for a burn-in=5000 23
3.16 3rd wave experiment: Sample iteration for the h parameter 23
3.17 3rd wave experiment: Density of the Γ parameter for a burn-in=5000 24
3.18 3rd wave experiment: Sample iteration for the Γ parameter 24

4.1 The initial state of our Tohoku simulation at t=0 26
4.2 The Tohoku simulation at any time t>0, the simulation presents unexpected NaN values 26

List of Tables

3.1 All parameters that affect our wave's initial shape and how they are correlated to each others 14
3.2 The first wave's unknown parameter and its real value 15
3.3 The second wave's experiments parameters 15
3.4 The third wave's experiments parameters 15
3.5 First wave results with use of only two probes 16
3.6 First wave results with use of four probes 16
3.7 Second wave experiment results 19

Bibliography

[1] N. Mori, T. Takahashi, T. Yasuda, and H. Yanagisawa. "Survey of 2011 Tohoku earthquake tsunami inundation and run-up". In: *Geophysical Research Letters* 38.7 (2011). DOI: 10.1029/2011GL049210. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2011GL049210. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2011GL049210.

[2] Y. Tanioka and K. Satake. "Tsunami generation by horizontal displacement of ocean bottom". In: *Geophysical Research Letters - GEOPHYS RES LETT* 23 (Apr. 1996), pp. 861–864. DOI: 10.1029/96GL00736.

[3] Y. Okada. "Surface deformation to shear and tensile faults in a halfspace". In: *Bulletin of the Seismological Society of America* 75 (Aug. 1985).

[4] R. M. Neal. "Probabilistic Inference Using Markov Chain Monte Carlo Methods". In: 1993.

[5] C. Andrieu, N. Freitas, A. Doucet, and M. Jordan. "An Introduction to MCMC for Machine Learning". In: *Machine Learning* 50 (Jan. 2003), pp. 5–43. DOI: 10.1023/A:1020281327116.

[6] A. Khan. "Seismogenic sources in the Bay of Bengal vis-A -vis potential for tsunami generation and its impact in the northern Bay of Bengal coast". In: *Natural Hazards* 61 (Apr. 2011). DOI: 10.1007/s11069-011-9970-x.

[7] F. Løvholt, G. Pedersen, and S. Glimsdal. "Coupling of Dispersive Tsunami Propagation and Shallow Water Coastal Response". In: *The Open Oceanography Journal* 4 (May 2010). DOI: 10.2174/1874252101004010071.

[8] A. Reinarz, D. E. Charrier, M. Bader, L. Bovard, M. Dumbser, K. Duru, F. Fambri, A.-A. Gabriel, J.-M. Gallard, S. Köppel, L. Krenz, L. Rannabauer, L. Rezzolla, P. Samfass, M. Tavelli, and T. Weinzierl. "ExaHyPE: An engine for parallel dynamically adaptive simulations of wave problems". In: *Computer Physics Communications* 254 (2020), p. 107251. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2020.107251. URL: http://www.sciencedirect.com/science/article/pii/S001046552030076X.

[9] H. Dierssen and A. Theberge. "Bathymetry: Assessing Methods". In: Jan. 2014, pp. 1–8. ISBN: 9781439852583. DOI: 10.1081/E-ENRW-120048588.

[10] M. B. Sebastian Rettenberger Meister Oliver. 2016. URL: https://www5.in.tum.de/pub/2016_easc_asagi.pdf.

[11] URL: https://doku.lrz.de/display/PUBLIC/Job+Processing+on+the+Linux-Cluster.

[12] I. M. von Würtemberg. "Ill-posed problems and their applications to climate research". In: 2011.

[13] M. Dashti and A. M. Stuart. "The Bayesian approach to inverse problems". In: *arXiv preprint arXiv:1302.6989* (2013).

[14] K.-R. Koch. "Bayes' Theorem". In: *Bayesian Inference with Geodetic Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 4–8. ISBN: 978-3-540-46601-7. DOI: 10.1007/BFb0048702. URL: https://doi.org/10.1007/BFb0048702.

[15] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev. "Why the Monte Carlo method is so important today". In: *WIREs Computational Statistics* 6.6 (2014), pp. 386–392. DOI: 10.1002/wics.1314. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.1314. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.1314.

[16] P. Paxton, P. Curran, K. Bollen, J. Kirby, and F. Chen. "Monte Carlo Experiments: Design and Implementation". In: *Structural Equation Modeling-a Multidisciplinary Journal - STRUCT EQU MODELING* 8 (Apr. 2001), pp. 287–312. DOI: 10.1207/S15328007SEM0802_7.

[17] J. S. Rosenthal. "Optimal Proposal Distributions and Adaptive MCMC". In: ().

[18] I. Yildirim. "Bayesian inference: Metropolis-hastings sampling". In: *Dept. of Brain and Cognitive Sciences, Univ. of Rochester, Rochester, NY* (2012).

[19] D. Vats, N. Robertson, J. M. Flegal, and G. L. Jones. "Analyzing MCMC output". In: *arXiv preprint arXiv:1907.11680* (2019).

[20] URL: http://muq.mit.edu/home.

[21] URL: https://github.com/annereinarz/MultilevelMCMCExaHyPE.