Coarse-to-fine MCMC in a seismic monitoring system

Xiaofei Zhou

December 4, 2015

Acknowledgement

I would like to express my sincere gratitude to my advisor Prof. Stuart Russell for the continuous support of my Master study, for his patience, motivation, and immense knowledge. His guidance helped me a lot in all the time of research and writing of this thesis.
Besides my advisor, I would like to thank David Moore, the 5th year PhD in our group, for his insightful comments and significant assistance during my research on this thesis.
Finally thanks physics department to allow me exploring my interests cross-disciplinarily outside of physics.

Coarse-to-fine MCMC in a seismic monitoring system

Xiaofei Zhou
Department of Electronic Engineering and Computer Science
Univeristy of California, Berkeley
Berkeley, CA, 94710
zhouxf92@berkeley.edu

December 1, 2015

Abstract

We apply coarse-to-fine MCMC to perform Bayesian inference for a seismic monitoring system. While traditional MCMC has difficulty moving between local optima, by applying coarse-to-fine MCMC, we can adjust the resolution of the model and this allows the state to jump between different optima more easily. It is quite similar to simulated annealing. We will use a 1D model as an example, and then compare traditional MCMC with coarse-to-fine MCMC and discuss the scaling behavior.

1 Introduction

Markov chain Monte Carlo (MCMC) [1] refers to a class of algorithms that sample from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. The state of the chain after a number of steps is used as a sample from the desired distribution. In theory, as the number of steps increases, the sample quality will improve. However, in practice, this method can often get stuck in local optimal of the distribution and fail to find all the modes of the target distribution. In this document, we will introduce coarse-to-fine MCMC to solve this problem.

Generally speaking, coarse-to-fine is a strategy that solves a problem first at a coarse scale and then at a fine scale, which lead to significant improvements in running time. Coarse-to-fine strategies have been applied to a wide range of problems. Marco [2] presents a method that can dramatically accelerate object detection with part based models. Fleuret [3] applies a coarse-to-fine sequential testing to localize all instances of a generic object class. There are also many related works on Markov decision process [4][5][6]. In this document, we will apply coarse-to-fine MCMC to deal with the seismic monitoring problem [7].

The layout of this paper is as follows:

We first introduce the necessary technical background and provide an introduction to Markov Chain Monte Carlo(MCMC) space. Then, we introduce 1D models including both a continuous model and discrete model. In the continuous model we assume the station can receive signals precisely, and the arrival time uncertainty is the key parameter we can adjust to control how coarse our model is. In the discrete model, we keep the arrival time uncertainty constant, and use the station resolution to control model coarseness. The continuous model is from NET-VISA [8], a detection based system and the discrete model is from SIG-VISA [9], a signal based system.

Next, we introduce parallel tempering, which is a critical step in coarse-to-fine MCMC [10] [11] after we have both the coarse and fine model. We run two or more independent Markov chains with different coarse degree simultaneously. Then we introduce swap moves which will swap the state between the chains. State in the coarse model jumps between optima more easily, so the swap move allows the fine chain to propose large jumps between modes for the fine chain.

Finally we show that coarse-to-fine lets us solve much larger problems under a given computational budget. Because the chain mixes faster, coarse-to-fine MCMC requires fewer steps. We show that this improves efficiency on a synthetic test dataset of seismic observations.

2 Technical background

In this chapter we review the necessary technical backgrounds. It covers the probability model for seismic monitoring, MCMC, and parallel tempering, which we use as the starting point for our coarse-to-fine MCMC.

2.1 Probability model

Suppose we have parameter, θ, and evidence, e. For each parameter θ we know the likelihood for generating evidence e is $p(e|\theta)$.

Usually the evidence e is what we can observe and θ is what we are interested in. We hope to infer parameter θ by evidence e, thus we want to know $p(\theta|e)$. Since we already have information for $p(e|\theta)$, we can infer $p(\theta|e)$ with Bayesian inference

$$p(\theta|e) \propto \frac{p(e|\theta) \times p(\theta)}{p(e)} \tag{1}$$

Where the likelihood $p(e|\theta)$ can be calculated. $p(\theta)$ is the prior. $p(e)$ is always constant because the evidence are given. In the case that the prior is a uniform distribution, the posterior probability is proportional to the likelihood.

Often the above calculation is intractable. In the next section, we will cover the Metropolis-Hasting algorithm, which allows one to efficiently sample from this distribution.

Another thing we need to notice is that due to different models which will be introduced later, $p(\theta|e)$ can have different shapes. The model with relative fat peak is called the "coarse version" while the model with sharp peak is called the "fine version".

2.2 Markov Chain Monte Carlo

In this section we use $\Pi(x)$ to represent $p(\theta|e)$. Now x is our notation for the state instead of θ. Suppose we need to sample from the distribution $\Pi(x)$. The core of the MCMC algorithm is to define a Markov Chain with $\Pi(x)$ as its stationary distribution.

The Metropolis-Hastings algorithm (MH) is a way to define a Markov chain with a particular target distribution. MH is defined by the target distribution, $\Pi(x)$, and a proposal distribution, $q(x \to x')$. A step of MH is defined by sampling a candidate next state $x' \sim q(x \to x')$. The Markov Chain then moves to x' with acceptance probability

$$r(x \to x') = min(1, \frac{\Pi(x')q(x' \to x)}{\Pi(x)q(x \to x')}) \tag{2}$$

The pseudo code is shown as

```
1. Initialize $x^{(0)}$
2. For $i = 0$ to $N-1$
       Sample $u \sim \mathcal{U}_{[0,1]}$
       Sample $x^* \sim q(x^{(i)} \to x^*)$
       If $u < r(x^{(i)} \to x^*)$
           next state is $x^{(i+1)} = x^*$
       else
           next state is $x^{(i+1)} = x^{(i)}$
```

Figure 1: Pseudocode for Metropolis-Hasting algorithm.

After we run the MH algorithm we have a sequence of states. If the iteration number is large enough, the histogram of the MCMC samples will approach the target distribution.

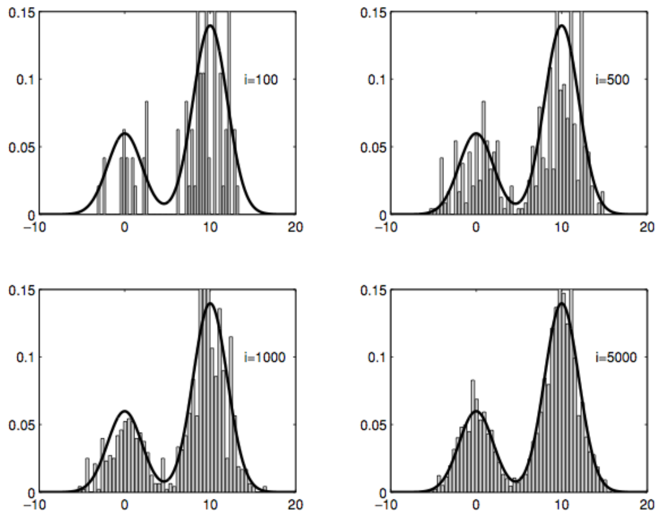

Figure 2: Target distribution and histogram of the MCMC samples at different iteration points.

The shape of the target distribution affects the rate of convergence as well as the frequency of state transfers between different modes. This is because the acceptance ratio is proportional to the slope of target distribution if the step size is constant. Usually the proposal distribution is symmetric, thus $q(x \to x') = q(x' \to x)$, so the acceptance rate has the following formula:

$$r(x \to x') = min\left(1, \frac{\Pi(x)}{\Pi(x')}\right) \tag{3}$$

For an “uphill” move, the acceptance rate is always 1. While for a “downhill” move, the fatter the peak is, the smaller the slope for a fixed size step, the smaller the difference between the posterior probability of two states, and thus the larger the acceptance rate.

Usually, there exists a crucial parameter controlling whether a model is coarse or fine, as we mentioned in previous section. Later, we will see that such parameters are similar to the annealing temperature in simulated annealing. In our model, it can be the arrival time uncertainty or just the resolution of the station.

Next we will introduce parallel tempering, which we will use to implement our coarse-to-fine MCMC algorithm.

2.3 Parallel tempering

Previously, we breifly introduced Markov Chains with different degree of coarse briefly. It is intuitive that this degree of coarseness can be viewed as temperature in simulated annealing.

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. The term annealing is from metallurgy and involves a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce defects. Simulated annealing interprets slow cooling as a slow decrease in the probability of accepting worse solutions as it explores the solution space. Accepting worse solutions is a fundamental property of metaheuristics because it allows for a more extensive search for the optimal solution.Thus, we know that the system will approach optima more quickly when we vary the temperature.

There are two common approaches to tempering: sequential tempering and parallel tempering. In sequential tempering, we adjust the parameter in the model during our algorithm. The coarse Markov chain will run first, and then the fine chain. This is quite similar to the simulated annealing in physics. In order to achieve the best configuration for a particular compound, we usually increase the temperature and then gradually cool it. Sometimes we repeat this several times.

Another method is called parallel tempering [12]. In parallel tempering, we maintain two separate chains: a coarse chain and a fine chain. They will have different target distributions, as illustrated below:

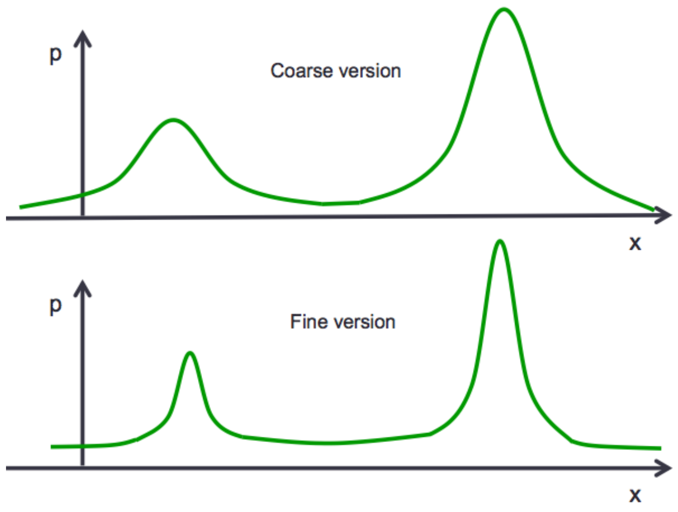

Figure 3: The coarse model has a relatively fat peak while the fine model has a relatively sharp peak

Since the MH acceptance rate depends on the slope of target distribution (or the height difference between peaks and valleys, and distance between peaks), the coarse chain will generally have a larger acceptance ratio compared to the fine chain. We can use the states in the coarse chain to "help" the states in the fine chain jump between different modes.

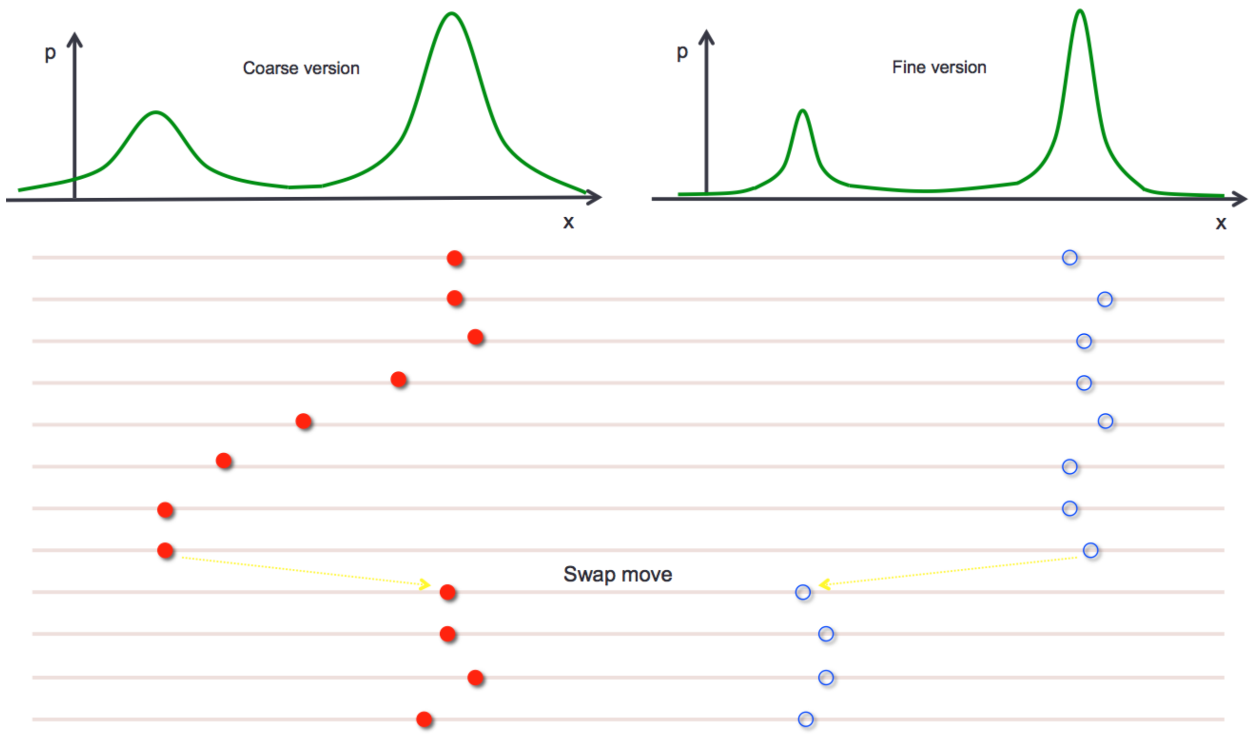

Figure 4: Swap move helps the state in the fine chain to jump between different optima. Red points represents the states in the coarse chain and blue hollow circles represents the states in the fine chain.

In parallel tempering, the coarse chain and fine chain chain run simultaneously. We use a swap move to switch the two states from different chains. We know the states in coarse chain are more likely to transfer between different modes compared with the states in fine chain. For example, in the picture above, the blue states are confined in one of the modes, while the red state can have larger probability to jump from one mode to another since its target distribution is fatter.

Thus, the probability that the state in the coarse chain jumps to another mode is relatively high. The swap move can switch these two states and the state in the fine chain will be in another mode. If we do not have a swap move, we expect to wait for a very long time for the state in the fine chain to jump from one mode to another.

Generally speaking, the swap move in coarse-to-fine MCMC will help the state in the fine chain jump between different modes more easily, and can solve much larger problems under a given computational budget.

3 1D continuous model

3.1 1D Continuous model description

We present these ideas in a 1D earthquake detection model. We have earthquake monitoring stations at $x = 0$ and $x = L$. Earthquakes occur in the internal $[0, L]$. For each earthquake event, we have parameter x and t describing the location and time for it. Once an event occurs, the earthquake wave will propagate towards the stations and stations will receive signals. Then, suppose we take a time period $[0, T]$, thus the state of each event are in the space $[0, L] \otimes [0, T]$.

Assume the velocity, v, of the wave is constant., thus it can be illustrated below:

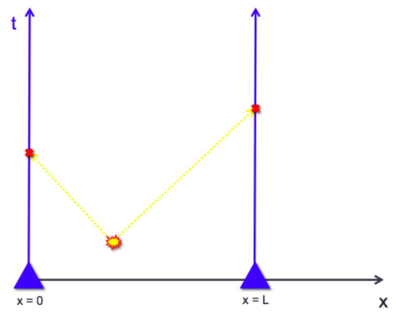

Figure 5: Illustration of 1D model. Once we have some event occurs, the station will receive signals from it.

However the station have uncertainty recording the arrival time with Gaussian variance σ. That means eventhough the theoretical arrival time for one event is $\bar{t}$, where $\bar{t}$ can be easily calculated from the information of station and event, the actual arrival time recorded by station is under Gaussian Distribution $\mathcal{N}(\bar{t}, \sigma^2)$.

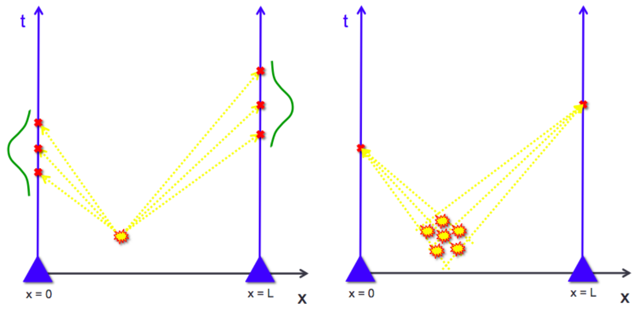

Figure 6: Because of the existence of arrival time uncertainty, for each event, the arrival time has a Gaussian distribution. And for each actual arrival time, there are many probable events corresponding to it.

Remember that the station signal e is our observation, and our goal is to infer the information of events θ.

3.2 Calculation of likelihood

3.2.1 Likelihood for only one event

In order to run Metropolis-Hastings, we need to calculate the likelihood, $p(e|\theta)$, for any state, θ, we are interested in. First, suppose the state only contains one event.

For the parameter θ of such event, assume the location of this event is x_1 and the time of this event is t_1. Thus, the event can be represented as (x_1, t_1). Remember that (x_1, t_1) is not the actual state(or event) which generated the actual observations: it is the state we want to get the likelihood of.

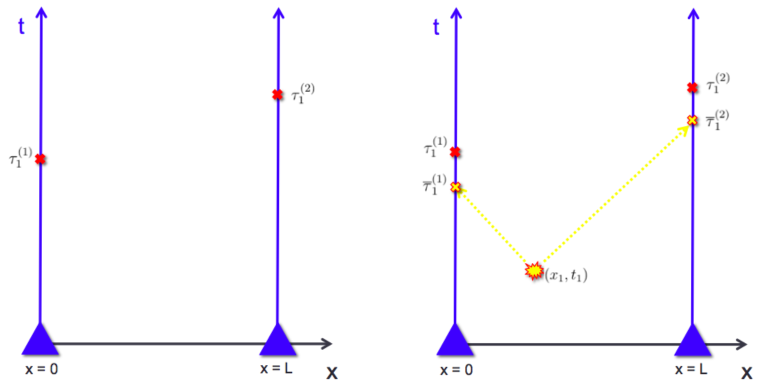

Figure 7: The left picture shows the evidence we observed. The evidence is the arrival times for two stations,$\tau_1^{(1)}$ and $\tau_1^{(2)}$, represented by the little red cross. The right picture shows for any state(x_1, t_1), we can predicted its arrival time$\tau_1^{(1)}$ and $\tau_1^{(2)}$, represented by the little yellow cross. Thus, we calculate the likelihood later.

For the evidence e, assume the first station s_1 at $x = 0$ receives the signal of such event at time $\tau_1^{(1)}$ where the superscript denotes the station and the subscript denotes which signal it received. Since we only have one event, the subscript is just 1. Similarily, assume the second station s_2 at $x = L$ receives the signal at time $\tau_1^{(2)}$. Thus we have

$$p(e|\theta) \doteq p(\tau_1^{(1)}, \tau_1^{(2)}|x_1, t_1). \tag{4}$$

For event (x_1, t_1), we can predict its arrival time at stations s_1 and s_2 if there doesn't exist any arrival time uncertainty. For station s_1, the distance between the event and s_1 is x_1. Thus the theoretical arrival time $\overline{\tau}_1^{(1)}$ should be $t_1 + x_1/v$. Similarly, for station s_2, the distance between the event and s_2 is $L - x_1$. Thus the theoretical arrival time $\overline{\tau}_1^{(2)}$ is $t_1 + (L - x_1)/v$.

However, the actual time for the arriving signal we received is not exactly the predicted one. For station s_1 it is $\tau_1^{(1)}$ and for station s_2 it is $\tau_1^{(2)}$. Thus the original likelihood can be expressed as

$$p(\tau_1^{(1)}, \tau_1^{(2)} | x_1, t_1) = p(\tau_1^{(1)}, \tau_1^{(2)} | \overline{\tau}_1^{(1)}, \overline{\tau}_1^{(2)}). \tag{5}$$

Where $\overline{\tau}_1^{(1)} = t_1 + x_1/v$ and $\overline{\tau}_1^{(2)} = t_1 + (L - x_1)/v$.

Since the signals at different stations are independent, we can rewrite the likelihood as

$$p(\tau_1^{(1)}, \tau_1^{(2)} | \overline{\tau}_1^{(1)}, \overline{\tau}_1^{(2)}) = p(\tau_1^{(1)}, | \overline{\tau}_1^{(1)}) \times p(\tau_1^{(2)} | \overline{\tau}_1^{(2)}). \tag{6}$$

Where $p(\tau_1^{(1)}, |\overline{\tau}_1^{(1)})$ is only related to the first station s_1 and $p(\tau_1^{(2)}|\overline{\tau}_1^{(2)})$ is only related to the second station s_2.

Since we model the actual arriving signal recorded by station using a Gaussian distribution, we can calculate $p(\tau_1^{(1)}, |\overline{\tau}_1^{(1)})$ and $p(\tau_1^{(2)}|\overline{\tau}_1^{(2)})$ as below:

$$\begin{aligned} p(\tau_1^{(1)}, |\overline{\tau}_1^{(1)}) &= \mathcal{N}(\tau_1^{(1)}, \overline{\tau}_1^{(1)}, \sigma) = \tfrac{1}{\sqrt{2\pi\sigma}} exp(-\tfrac{(\tau_1^{(1)} - \overline{\tau}_1^{(1)})^2}{2\sigma^2}) \\ p(\tau_1^{(2)}, |\overline{\tau}_1^{(2)}) &= \mathcal{N}(\tau_1^{(2)}, \overline{\tau}_1^{(2)}, \sigma) = \tfrac{1}{\sqrt{2\pi\sigma}} exp(-\tfrac{(\tau_1^{(2)} - \overline{\tau}_1^{(2)})^2}{2\sigma^2}). \end{aligned} \tag{7}$$

Thus, we know the likelihood is

$$p(e|\theta) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(\tau_1^{(1)} - \overline{\tau}_1^{(1)})^2}{\sigma^2}) \times \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(\tau_1^{(2)} - \overline{\tau}_1^{(2)})^2}{2\sigma^2}).$$

It can be expressed as

$$p(e|\theta) = \prod_{k=1}^{2} \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(\tau_1^{(k)} - \overline{\tau}_1^{(k)})^2}{2\sigma^2}). \tag{8}$$

Where k is the index for the station.

3.2.2 Likelihood for multiple events

Now we generalize to multiple events. Suppose we have n events, each can be described as $(x_1, t_1), \ldots (x_n, t_n)$. For the first station s_1, the arrival time of the signals it received is $(\tau_1^{(1)}, \ldots, \tau_n^{(1)})$. For the second station s_2, the arrival time is $(\tau_1^{(2)}, \ldots, \tau_n^{(2)})$.Then, the likelihood can be expressed as

$$p(e|\theta) = \prod_{k=1}^{2} \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(\tau_i^{(k)} - \overline{\tau}_i^{(k)})^2}{\sigma^2}) \tag{9}$$

Where k is the index for the station and i is the index for the events.

Note that when the number of events increase, there may exist more than one probable solution. In fact, as the dimension our state increases, there will occur more local modes. High dimension is hard to visualize, let's just use the two events case as an example.

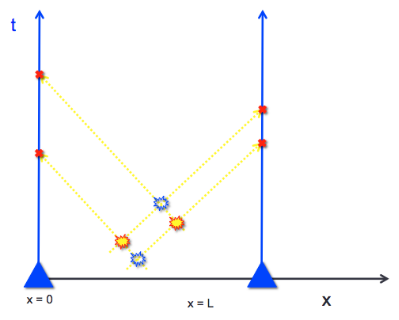

Figure 8: Use two events in the1D model as an example. Both yellow states and blue hollow states can give the same observation. Thus we say that we have two modes. In reality, different modes may have different probability distribution shapes, which might cause that some optimum state is more likely to occure than other optima.

Generally speaking, the posterior probability contains many modes, and each mode is a multivariate Gaussian distribution.

3.3 Arrival time uncertainty

It is obvious that the arrival time uncertainty, σ, is a crucial parameter because it will affect the posterior probability distribution. If σ is large then the peak of the optima in the posterior would be quite fat, while for small σ the peak in the posterior would be sharp.

This is illustrated in the picture shown as below:

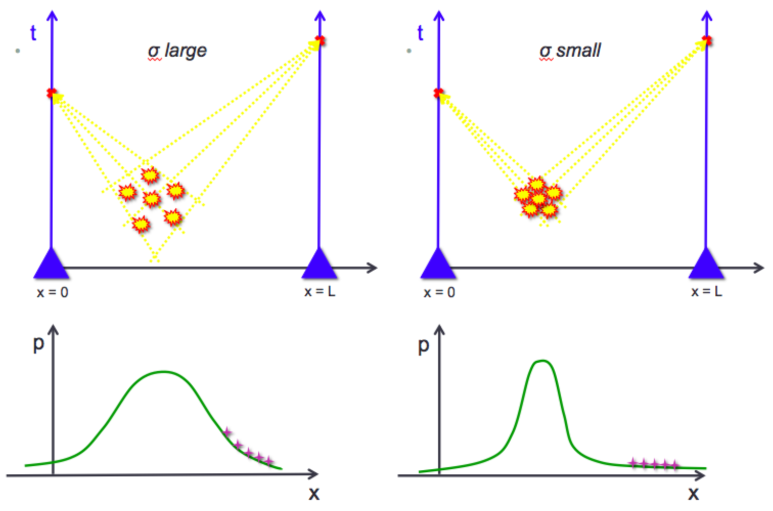

Figure 9: The left part is the model with a large arrival time uncertainty while the right part is the model with a small uncertainty.

We can see the model with a small arrival time uncertainty will have a quite sharp posterior probability distribution. Thus the slope is very small for most of the region. We say this is a fine version of the model compared to the coarse version model.

3.4 Proposal move

In previous section, we introduced both the coarse and fine versions of the model , as well as the calculation of likelihood. In this section we need to figure out the proposal moves so that we can implement our parallel tempering algorithm.

We have two kinds of proposal moves. The first one is a normal move that changes the position of the state in one chain. The second is called a swap move that can switch the state between the coarse and fine chain. In this section we use "state" to denote the state in one chain, and use "parallel state" to denote the combination of states from different chains.

3.4.1 Normal move

In this $1D$ case, our state space is $x - t$ space. Suppose we propose a new state under a Gaussian distribution.

$$\begin{aligned} q(\theta \to \theta') &= q(x, t \to x', t') \\ &= \mathcal{N}(x|x', \sigma_{pps\_x})\ \mathcal{N}(t|t', \sigma_{pps\_t}) \end{aligned} \tag{10}$$

Where σ_{pps_x} and σ_{pps_t} are the standard deviations. It shows how much each move makes the new state deviate from the original.

This is reasonable and is symmetric. So $q(\theta \rightarrow \theta') = q(\theta' \rightarrow \theta)$. Then, the acceptance rate can be expressed as:

$$\begin{aligned} r(\theta \rightarrow \theta') &= min\left(1, \frac{\Pi(\theta)q(\theta \rightarrow \theta')}{\Pi(\theta')q(\theta' \rightarrow \theta)}\right) \\ &= min\left(1, \frac{\Pi(\theta)}{\Pi(\theta')}\right) \end{aligned} \tag{11}$$

Now, let's run the MH algorithm in one chain first as an example.

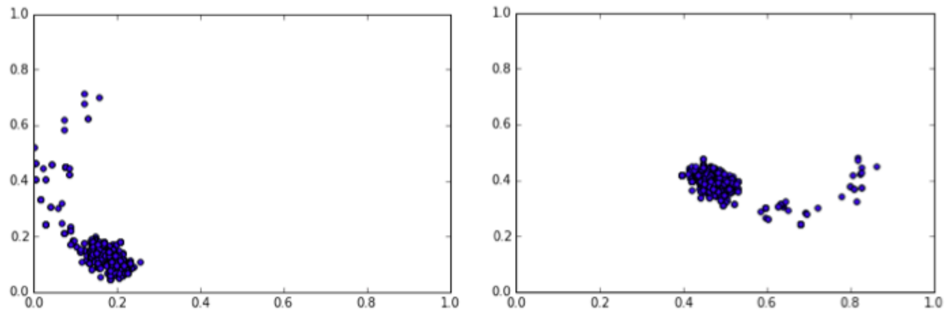

Figure 10: An example of MH algorithm. Suppose we have two events with $[x_1, t_1] = [0.17, 0.13]$ and $[x_2, t_2] = [0.49, 0.38]$. We can see the state will jump from an random initial state to the actual state. And finally converge under the posterior probability distribution.

We can see that the initial state is far away from the peak of target distribution. Then it moves to the peak gradually and finally the sample distribution will converge to the target distribution.

In our model we assume the number of events is fixed. When the number of events is unsure, this number becomes a parameter of our state. Then, we will include birth and death moves in the proposal, which allow the dimension of state to vary during the MH algorithm [13][14].

3.4.2 Swap move

Now let's consider the parallel state (θ_1, θ_2) where θ_1 is the state in coarse chain and θ_2 is the state in fine chain. Thus, for the proposal distribution of swap move we have

$$q(\theta_1, \theta_2 \rightarrow \theta'_1, \theta'_2) = \begin{cases} 1 & \text{if } \theta_1 = \theta'_2, \theta_2 = \theta'_1 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Where the parameters, θ_1, in coarse chain are the same as the parameters, θ'_2, in the fine chain. Parameters, θ_2, in the fine chain are the same as the parameters, θ'_1, in the coarse chain.

Since it is symmetric we can rewrite the acceptance rate as

$$r(\theta_1, \theta_2 \to \theta_1', \theta_2') = min\left(1, \frac{\Pi(\theta_1, \theta_2)q(\theta_1, \theta_2 \to \theta_1', \theta_2')}{\Pi(\theta_1', \theta_2')q(\theta_1', \theta_2' \to \theta_1, \theta_2)}\right)$$
$$= min\left(1, \frac{\Pi(\theta_1, \theta_2)}{\Pi(\theta_1', \theta_2')}\right) \tag{13}$$

Where $\Pi(\theta_1, \theta_2) = \Pi(\theta_1)\ \Pi(\theta_2)$ is the probability of the parallel state is the product of the probability of those states in each chain.

In our algorithm, we run two Markov chains simultaneously. The swap move is only useful when the state in the coarse and fine chains both reach the mode. Thus, the swap move should be rare. Most of the proposal moves should be the normal move which change the position of the state in each chain.

Thus, for each step of MH algorithm, we say that we have a probability of $r_n = 0.99$ that we choose a normal move for both the coarse and fine chain, and a probability of $1 - r_n$ that we choose a swap move. r_n is a parameter we can adjust during our algorithm.

3.5 Results

In reality, we usually draw log probability of the state versus step to observe its convergence behavior. The state is usually high dimensional, so we draw one parameter of the state versus the step. We can see that the parameter will converge to and jump between those modes.

In our program, we set the length scale $L = 1$, time scale $T = 1$, velocity $v = 1$, standard deviation of arrival time uncertainty $\sigma_{coarse} = 0.2$ for coarse chain and $\sigma_{fine} = 0.05$ for fine chain. The standard deviation of proposal distribution is $\sigma_{pps\_x} = \sigma_{pps\_t} = 0.02$

We set our actual state to be $(x_1, t_1) = (0.3, 0.5)$, $(x_2, t_2) = (0.7, 0.5)$, thus our target distribution has two modes.

First, we only run the fine chain. We run 50000 steps and plot the state parameter against the step:

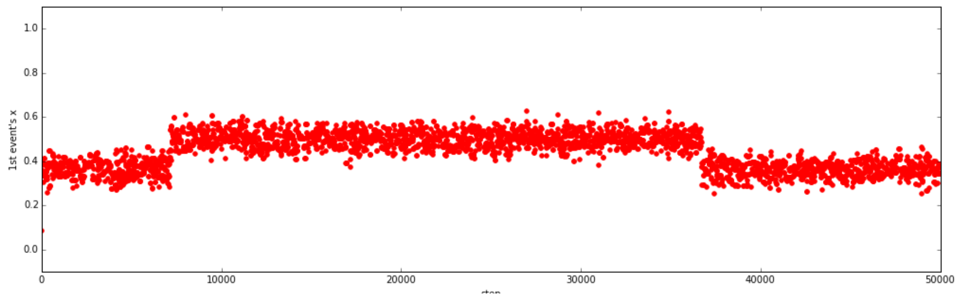

Figure 11: This picture shows state parameter v.s. steps for fine chain with $\sigma = 0.05$. In this case, it is x_1 v.s. steps. We also have t_1, x_2, t_2 v.s. steps but we don't draw it here since the parameters of one of the events are already enough to show the result. From the picture we see that for the fine chain, it is rare for one state to jump between modes. In 10000 steps there are only two jumps.

From the picture above we can see that if we run standard MCMC, the frequency for a state transferring between different modes is very slow.

Then we run the parallel tempering program. The results are listed below

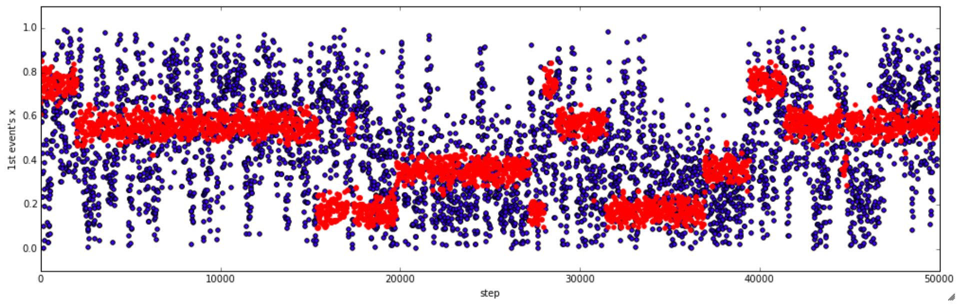

Figure 12: This picture shows the case for parallel tempering. The red line denotes the fine chain with $\sigma = 0.05$ and blue line denotes the coarse chain with $\sigma = 0.2$. By introducing the swap move, we can see that the state in fine chain jumps between different modes frequently. This is because state in coarse chain is more easily to jump from one modes to another, so the swap move can switch the fine state from one mode to another.

We can see that both the blue state and red state jump between different modes frequently. The frequency for coarse-to-fine MCMC is about 5 times larger than traditional MCMC.

Next, we draw the graph of log probability v.s. steps to see that coarse-to-fine MCMC converges quicker than traditional MCMC

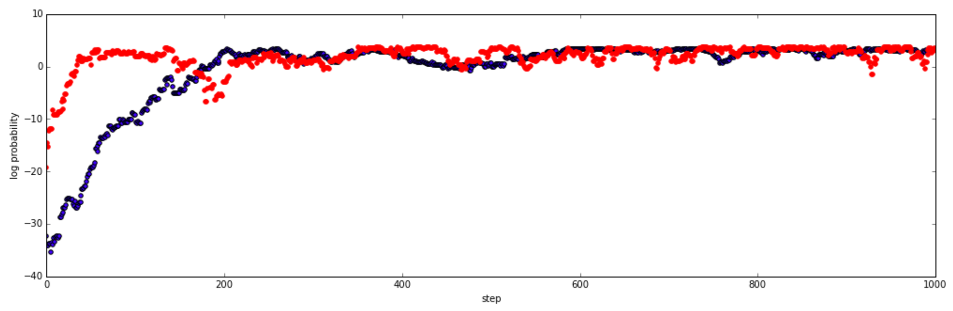

Figure 13: This picture shows the log probability versus steps for coarse-to-fine MCMC(red) and traditional MCMC(blue). We can see coarse-to-fine MCMC converges quicker than traditional MCMC.

From above two pictures we see that coarse-to-fine MCMC converges to one mode and switches between different modes much quicker than traditional MCMC, while providing the same precision.
Finally, we draw the graph of autocorrelation v.s. lags to compare the performance between coarse-to-fine MCMC and traditional MCMC.

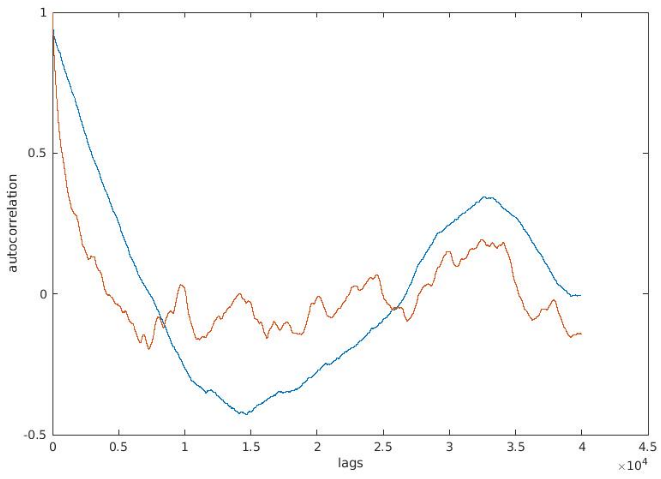

Figure 14: This picture shows the autocorrelation v.s. lags for coarse-to-fine MCMC(red) and traditional MCMC(blue). We can see the autocorelation of coarse-to-fine MCMC converges to zero much quicker than traditional MCMC, which means the state is easier to jump to other positions after a long enough steps.

The picture above shows autocorrelation of state parameter v.s. lags. When lag increases, the autocorrelation approaches to zero. This means that a state's position is unrelated to its position after enough steps. We can see that, for coarse-to-fine MCMC, it converges quicker than traditional MCMC.

These results show that coarse-to-fine MCMC performs better.

4 1D discrete model

4.1 Insight of introducing discrete model

In the previous model, adjusting the arrival time uncertainty controls how coarse our continuous model is. Now, we introduce the discrete model which is closer to the signal-based system.

Suppose each station has a resolution and it can not give the precise arrival time in a particular time period. But it can tell you how many signals it received during that period. When the resolution is low, we define it as “coarse”. Compared to the continuous model, arrival time uncertainty as a measure of “coarseness” is just an analogy for discretization. Similarly, when the resolution is low, it corresponds to a fine version model.

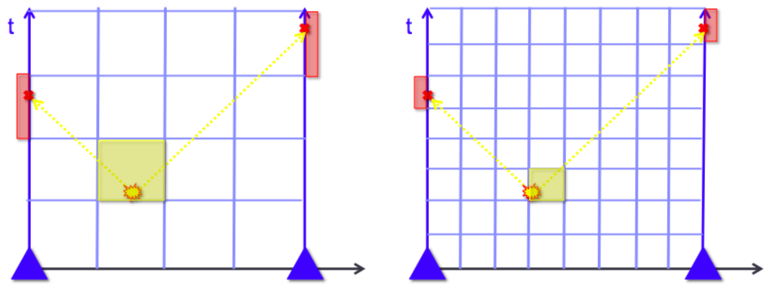

Figure 15: Suppose now we have discretized model. The left part corresponds to a coarse chain while the right part corresponds to a fine chain.

By introducing this new parameter resolution, we can apply coarse-to-fine MCMC without modifying the seismic model.

4.2 Discrete model description

4.2.1 Time scale for stations

In order to introduce the discrete model, let’s first consider the time scale for event. In the continuous model, the time scale for the events is $[0, T]$, thus the time scale for the station is $[0, T + L/v] = [0, T_s]$.

4.2.2 Station resolution

In the continuous case, the arrival time recorded by each station is a real number. However, in the discrete case, we suppose each station has finite resolution and the actual arrival time can not be very precise. But it can tell you how many signals it received in this period. We say the station has high resolution when the time period is short and low resolution when the time period is long.

For example, suppose the time scale of the station is $[0, 1]$. Then if the resolution of the station is 0.1, we have 10 possible choices for the arrival time. If the resolution is 0.01 then we have 100 choices, and similarly, 1000 choices for resolution 0.001.

After we set the station resolution, we can see the arrival time of the events are discretized. It has only finite states. Suppose the resolution, res, of the station is $T_s/5$, thus the time periods are $[0, T_s/5]$, $[T_s/5, 2T_s/5]$, $[2T_s/5, 3T_s/5]$, $[3T_s/5, 4T_s/5]$, $[4T_s/5, T_s/5]$. We represent it as τ_1, τ_2, τ_3, τ_4, τ_5. Remember that in discrete model τ represents a particular time period instead of a precise time point.

4.2.3 Signal amplitude

Next, let's assume each event contains some units of energy, represented by ene. When the wave of event arrives at the station, the average magnitude of the signal is proportional to ene/res. This is intuitive since, if the resolution is fine, then the station will receive this energy in a smaller time period and enhance its magnitude level.

As an example, for station s_1, suppose one arriving signals come during time period τ_2 and two signals come during time period τ_4. The signal received should be

$$signal^{(1)} = [0, \frac{ene}{res}, 0, 2\frac{ene}{res}, 0]. \tag{14}$$

The signal can be illustrated below:

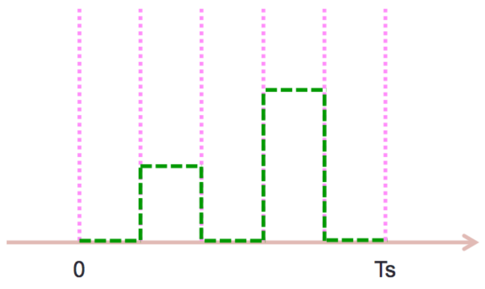

Figure 16: T_s is the time scale for the station. Suppose the resolution is $T_s/5$. Then, the green dotted line represents the signal the station received when τ_2 receives one arriving signal and τ_4 receives two arriving signals.

4.2.4 Noise

Because seismic signals are noisy, we include noise in our model. In addition, by including noise, we can directly apply a Gaussian distribution to calculate the likelihood, which can avoid unrealistic delta functions.

Suppose we have noise in our signal with mean μ and variance σ_{noise}. For each period, τ_i, the amplitude level of noise, no_i, is under $\mathcal{N}(\mu, \sigma_{noise})$.

Thus we can express the signal as the formula below when we consider noise:

$$signal^{(1)} = [no_1, no_2 + \frac{ene}{res}, no_3, no_4 + 2\frac{ene}{res}, no_5]. \tag{15}$$

The signal can be illustrated below:

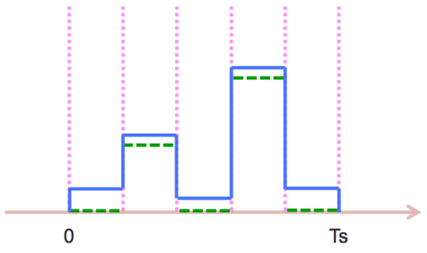

Figure 17: The green dotted line is the signal without noise, while the blue solid line is the actual signal when we consider noise.

Thus, for a particular resolution, our observation is one such signal for every station. We hope to infer the actual events.

4.3 Calculation of likelihood

4.3.1 Likelihood for only one event

For simplicity, let's consider the case with only one event first. The evidence for one station is like

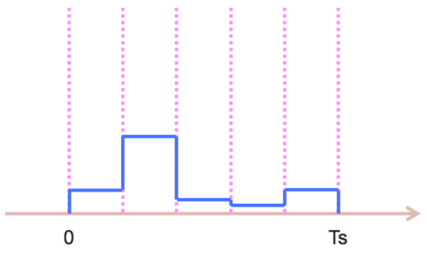

Figure 18: The actual signal above shows the case of only one events. Notes that in each period, τ_i, the magnitude level is non-zero because noise exists.

We express this signal as $si^{(k)} = [si_1^{(k)}, \ldots, si_5^{(k)}]$ where k represents the index of station. Then, for any given state (x_1, t_1), the likelihood can be expressed as

$$p(e|\theta) = p(si^{(1)}, si^{(2)}|x_1, t_1) \tag{16}$$

For a particular state (x_1, t_1), we can predicted its arrival time for a given station, $\overline{\tau}_1^{(k)}$, where k represents the index of the station. Suppose $\overline{\tau}_1^{(k)}$ arrives in τ_2. In this case, it can illustrated in the picture below:

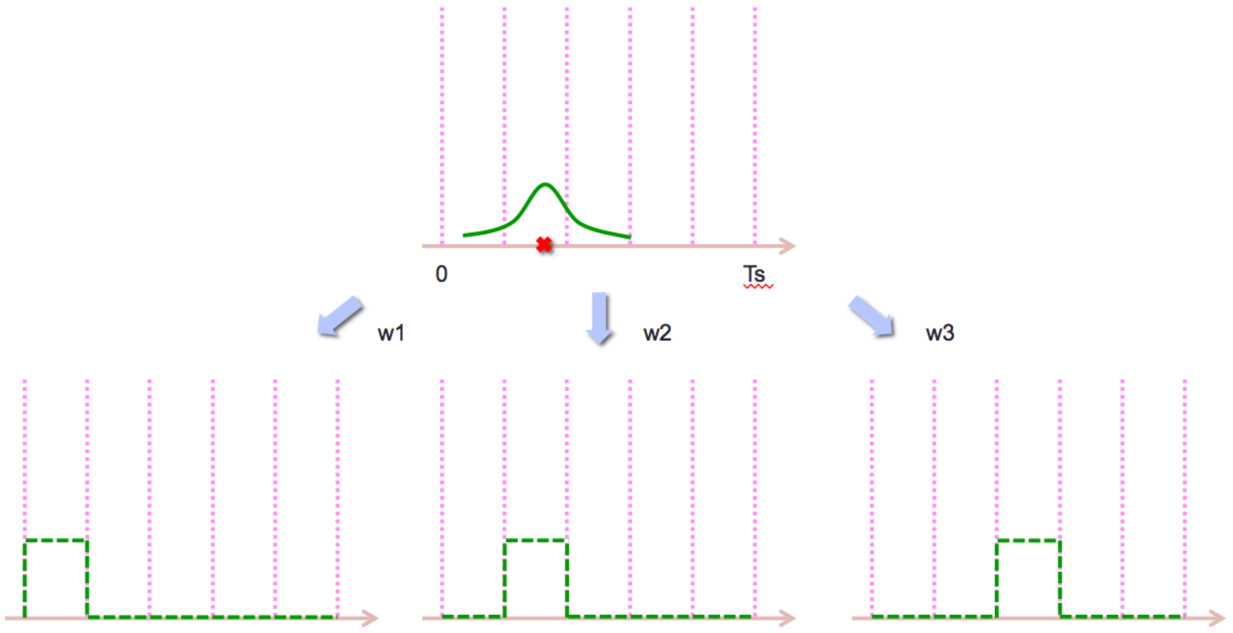

Figure 19: For any given state we can predict its arrival time presented by the little red cross. Due to the arrival time uncertainty it may corresponds to multiple predicted signals with particular weights.

Due to the arrival time uncertainty, these arrival times are located in adjacent time periods with some set probability. Though it should be located in τ_2 theoretically, it can also be located τ_1 and τ_3.

Thus, we need to calculate the weights for predicted arrival times being located in each particular τ_i. Such weight in fact is the integral of a Gaussian function in each time period. In the example above, we assume the probability that τ_1 receives a signal is w_1, similarly the probabilities for τ_2 and τ_3 are w_2 and w_3. Because the Gaussian distribution has an infinitly long tail, we cut the tail at two deviations to simplify the calculation. Remember that we need to do renormalization for w_i during the calculation. After that we can see for each predicted signal we have a weight w_i that corresponds to it:

$$w_i = \int_{\tau_i} \mathcal{N}(t|\overline{\tau}_1^{(k)}, \sigma)dt. \tag{17}$$

Where $\overline{\tau}_1^{(k)}$ is the predicted arrival time.

For each w_i ($i = 1, 2, 3$), suppose the predicted signal is $pred\_si^{(k,i)}$, then the likelihood can be expressed as

$$p(e|\theta) = \sum_{i=1}^{3} w_i \times p_i(si^{(1)}, si^{(2)}|pred\_si^{(1),i}, pred\_si^{(2),i}) \tag{18}$$

Where i denotes different cases and p_i denotes the likelihood for each case. Since the signals in different stations are independent, we can rewrite the likelihood as:

$$\begin{aligned} p(e|\theta) &= \sum_{i=1}^{3} w_i \times p_i^{(1)}(si^{(1)}|pred\_si^{(1),i}) \times p_i^{(2)}(si^{(2)}|pred\_si^{(2),i}) \\ &= \sum_{i=1}^{3} w_i \prod_{k=1}^{2} p_i^{(k)}(si^{(k)}|pred\_si^{(k),i}) \end{aligned} \tag{19}$$

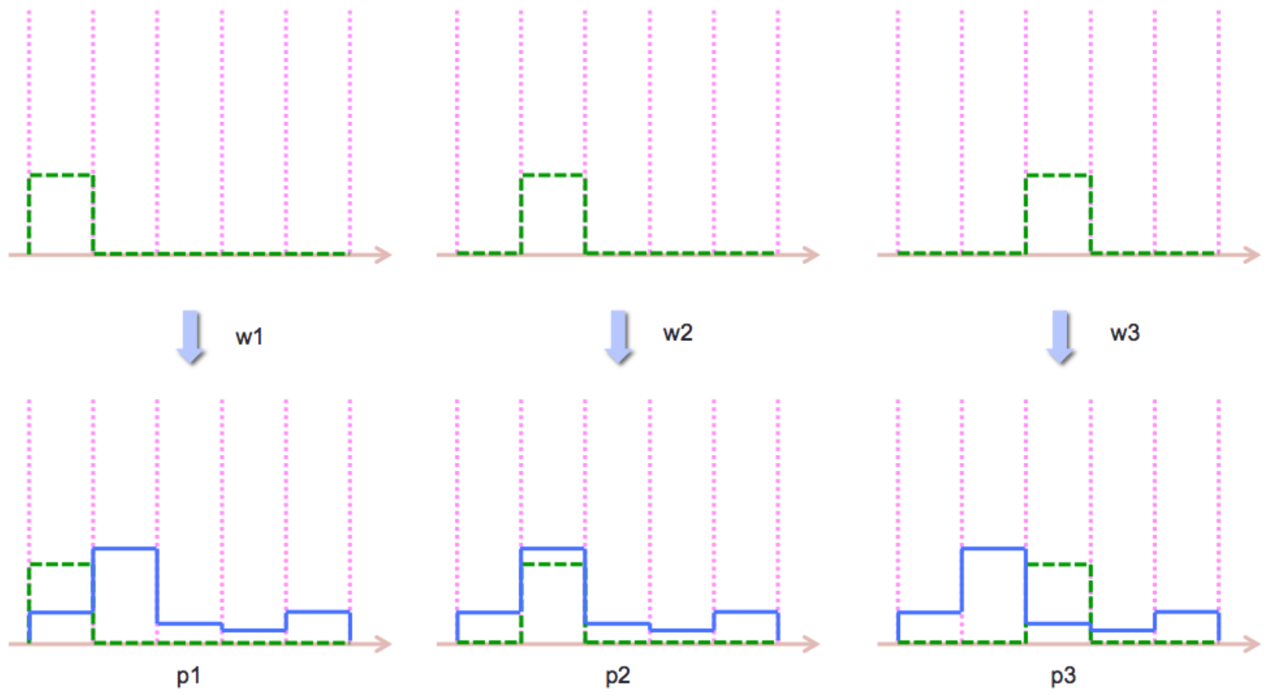

Figure 20: For a given state and station we can calculate its w_i and likelihood p_i for each case i. The likelihood is the sum for each case, $\sum_i w_i \times p_i$.

Next, for each case w_i ($i = 1, 2, 3$), we can compare the actual signal with the predicted signal and then calculate its likelihood p_i. From previous part we know the signal consists of 5 parts since the resolution is $T_s/5$. Thus for any actual signal $si^{(k)}$ and predicted signal $pred\_si^{(1),i}$, we have:

$$si^{(k)} = [si_1^{(k)}, \ldots, si_5^{(k)}] \tag{20}$$

$$pred\_si^{(k),i} = [pred\_si_1^{(k),i}, \ldots, pred\_si_5^{(k),i}] \tag{21}$$

Since the magnitude level in different periods are independent, and in each period, the difference of the actual magnitude level and predicted magnitude level is due to the uncertainty of noise, we know the smaller difference between the actual magnitude level and predicted actual level, the higher probability that this prediction is true.

Thus, we can express each likelihood as

$$p_i^{(k)}(si^{(k)}|pred\_si^{(k),i}) = \prod_{j=1}^{5} \mathcal{N}(si_j^{(k)}, pred\_si_j^{(k),i}, \sigma_{noise}). \tag{22}$$

Finally, the full likelihood is expressed as:

$$\begin{aligned} p(e|\theta) &= p(si^{(1)}, si^{(2)}|x_1, t_1) \\ &= \sum_{i=1}^{3} w_i \prod_{k=1}^{2} \prod_{j=1}^{5} \mathcal{N}(si_j^{(k)}, pred\_si_j^{(k),i}, \sigma_{noise}). \end{aligned} \tag{23}$$

Where i denotes any probable cases, k denotes different stations, j denotes to different time period for the station signal.

This is only for the state with one event, thus i is relatively small. Later we will see that when the number of events increase, the probable cases for the state will increase exponentially.

4.3.2 Likelihood for multiple events

The likelihood for multiple events are similar. Use two events as an example.

Suppose for some state, two predicted arrival time locates in τ_2 and τ_4. Due to the arrival time uncertainty, for the first arrival, it can be located either in τ_1, τ_2, τ_3. And for the second arriving, it can be located either in τ_3, τ_4, τ_5. Thus we have $3 \times 3 = 9$ cases which can be shown in the picture below.

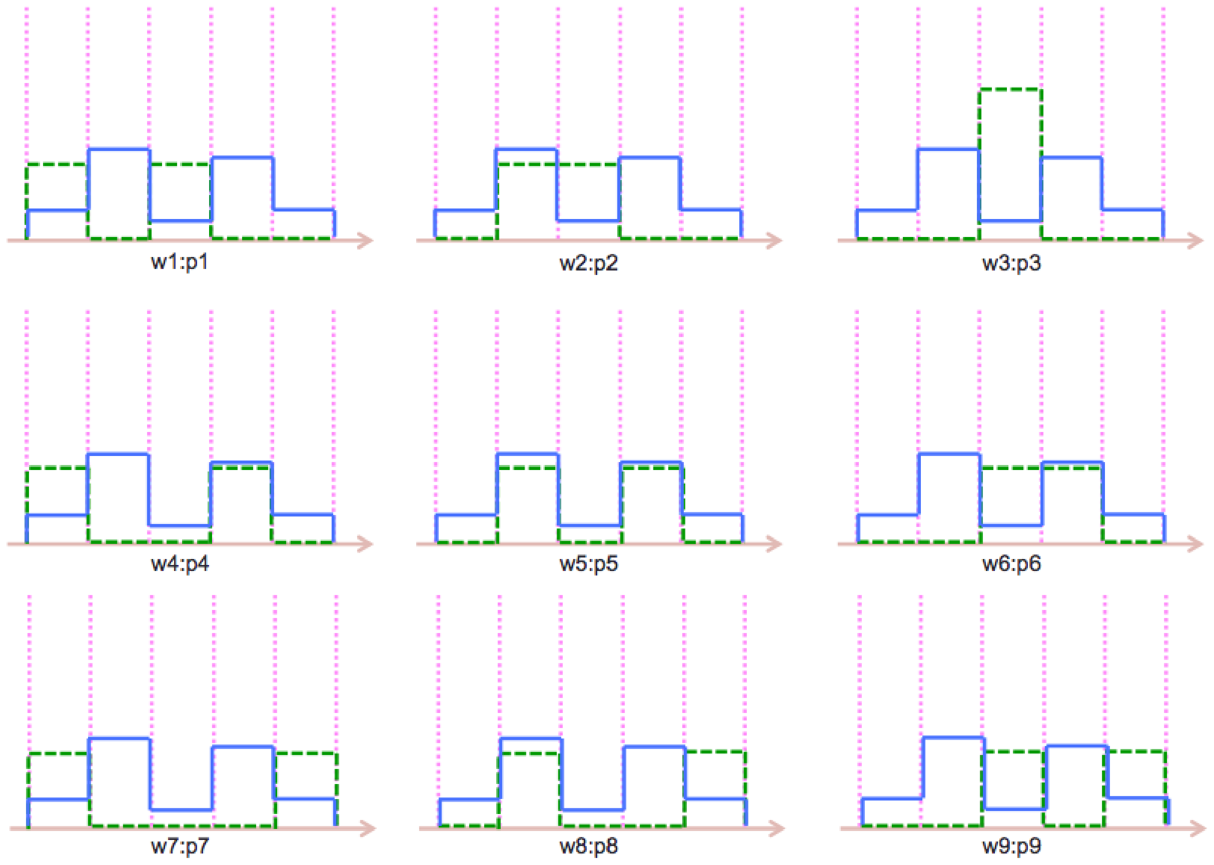

Figure 21: The picture above shows 9 cases of probable predicted signals for two events state and compare it with the actual signal. For the same column, the first arriving signal are always in the same τ, while different row means the second arriving signal are in different τ. Similarily, for the same row, the second arriving signal are always in the same τ, while different columns mean that the first arriving signal are in different τ. The green dotted line is predicted signal and blue solid line is actual signal. The resolution of the station is $T_s/5$.

From the above picture, we can see in most of the cases (except w_5), the overlap of the predicted signal and actual signal is quite small. The most probable case is w_5. The cases far away from w_5 have little probability.

Thus, the likelihood is

$$p(e|\theta) = \sum_{i=1}^{3} w_i \prod_{k=1}^{2} \prod_{j=1}^{5} \mathcal{N}(si_j^{(k)}, pred\_si_j^{(k),i}, \sigma_{noise}). \tag{24}$$

It is the same as the likelihood in the case of a single event, while w_i represents all the cases here.

4.4 Proposal move

In previous chapter, we introduced the proposal move for the continuous model. It is similar for the discrete model. [7]

For a normal move

$$q(\theta \to \theta') = \mathcal{N}(x, x', \sigma_{pps\_x})\ \mathcal{N}(t, t', \sigma_{pps\_t}). \tag{25}$$

Thus, the acceptance rate is

$$r(\theta \to \theta') = min\left(1, \frac{\Pi(\theta)}{\Pi(\theta')}\right). \tag{26}$$

For a swap move

$$q(\theta_1, \theta_2 \to \theta'_1, \theta'_2) = \begin{cases} 1 & \text{if } \theta_1 = \theta'_2, \theta_2 = \theta'_1 \\ 0 & \text{otherwise} \end{cases}. \tag{27}$$

Thus the acceptance rate is

$$r(\theta_1, \theta_2 \to \theta'_1, \theta'_2) = min\left(1, \frac{\Pi(\theta_1, \theta_2)}{\Pi(\theta'_1, \theta'_2)}\right) \tag{28}$$

4.5 Results

For the discrete model, the results are quite similar with those of the continuous model. In our discrete model, we set the length scale $L = 1$, time scale $T = 1$, velocity $v = 1$, standard deviation of arrival time uncertainty $\sigma = 0.1$. The resolution of station for coarse model is 0.4 and the resolution for fine model is 0.2. The standard deviation of proposal distribution is $\sigma_{pps\_x} = \sigma_{pps\_t} = 0.02$

We set our actual state to be $(x_1, t_1) = (0.3, 0.5)$, $(x_2, t_2) = (0.7, 0.5)$, thus our target distribution has two modes. It is the same as the continuous model.

First, we only run the fine chain. We run 50000 steps and draw state parameter v.s step diagram.

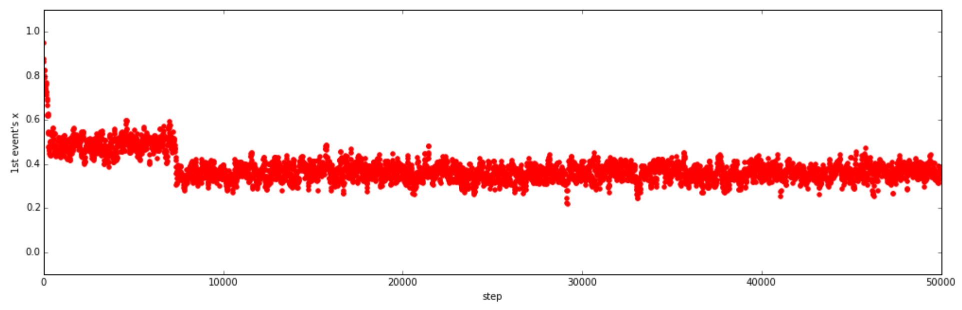

Figure 22: This picture shows state parameter v.s. steps for fine chain with $res = 0.2$. We see that for the fine chain, it is rare for one state jumping between different modes. In 10000 steps there is only one jump.

Thus, we know that, in standard MCMC, the frequency for a state transferring between different modes is low.

Then, we run the parallel tempering program.

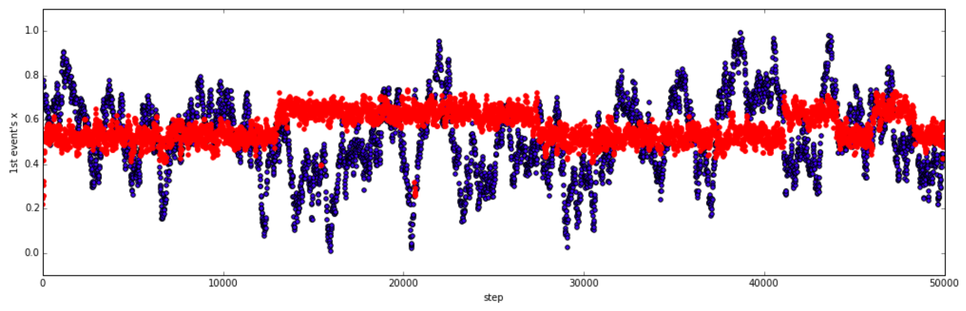

Figure 23: This picture shows the case for parallel tempering. The red line denotes the fine chain with $res = 0.2$ and the blue line denotes the coarse chain with $res = 0.4$. By introducing the swap move, we can see that the state in the fine chain jumps between different modes frequently.

We can see that both the blue states and red states jump between different modes frequently.

Finally, we draw the graph of autocorrelation v.s. lags to compare the performance between coarse-to-fine MCMC and traditional MCMC.

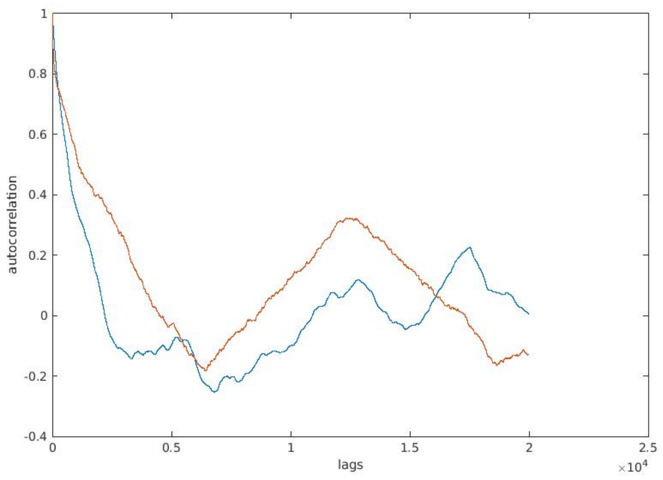

Figure 24: Autocorrelation v.s. lags for coarse-to-fine MCMC(blue) and traditional MCMC(red). The autocorelation of coarse-to-fine MCMC converges to zero much quicker than traditional MCMC.

Generally speaking, the results for the discrete model are similar to the results from the continuous model. We can say that coarse-to-fine MCMC has better performance in both of these models. It provides shorter convergence time as well as higher frequency jumps between modes.

5 Conclusion

The coarseness of a chain is crucial for the behavior of sampling in MCMC. Coarse chains provide more opportunities to jump between different modes. While fine chains give more precise solution for a particular mode.

Coarse-to-fine MCMC combines the advantage of both of them. It mixes the chains with different coarseness and helps the state in fine chain to jump between different modes as quickly as those states in coarse chain. It gives very precise solutions for one mode, and explores different modes at the same time.

In the previous examples, we compared how states jump between mode in coarse-to-fine and traditional MCMC. We conclude that coarse-to-fine MCMC has a higher frequency for state transferring between different modes. We also compared log probability and autocorrelation to confirm that. Thus, in order to achieve the same result, coarse-to-fine MCMC requires fewer steps. The efficiency would be enhanced for such problems.

References

[1] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[2] Marco Pedersoli, Andrea Vedaldi, and Jordi Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1353–1360. IEEE, 2011.

[3] Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1-2):85–107, 2001.

[4] Jake Bouvrie and Matteo Maggioni. Efficient solution of markov decision problems with multiscale representations. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 474–481. IEEE, 2012.

[5] Dave Higdon, Herbert Lee, and Zhuoxin Bi. A Bayesian approach to characterizing uncertainty in inverse problems using coarse and fine-scale information. *Signal Processing, IEEE Transactions on*, 50(2):389–399, 2002.

[6] Yali Amit, Donald Geman, and Xiaodong Fan. A coarse-to-fine strategy for multiclass shape detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(12):1606–1621, 2004.

[7] Nimar Arora, Stuart J Russell, Paul Kidwell, and Erik B Sudderth. Global seismic monitoring as probabilistic inference. In *Advances in Neural Information Processing Systems*, pages 73–81, 2010.

[8] Nimar S Arora, Stuart Russell, and Erik Sudderth. NET-VISA: Network processing vertically integrated seismic analysis. *Bulletin of the Seismological Society of America*, 103(2A):709–729, 2013.

[9] David A Moore, Kevin M Mayeda, Steve M Myers, Min Joon Seo, and Stuart J Russell. Progress in signal-based Bayesian monitoring. *Proceedings of the 2012 monitoring research review: ground-based nuclear explosion monitoring technologies, LA-UR-12-24325*, pages 263–273, 2012.

[10] Yalchin Efendiev, Thomas Hou, and Wuan Luo. Preconditioning Markov chain Monte Carlo simulations using coarse-scale models. *SIAM Journal on Scientific Computing*, 28(2):776–803, 2006.

[11] Yalchin Efendiev, Bangti Jin, Presho Michael, and Xiaosi Tan. Multilevel Markov chain Monte Carlo method for high-contrast single-phase flow problems. *Communications in Computational Physics*, 17(01):259–286, 2015.

[12] Charles J Geyer. Markov chain Monte Carlo maximum likelihood. 1991.

[13] Songhwai Oh, Stuart Russell, and Shankar Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 735–742. IEEE, 2004.

[14] Peter J Green and David I Hastie. Reversible jump MCMC. *Genetics*, 155(3):1391–1403, 2009.