PARALLEL MARKOV CHAIN MONTE CARLO METHODS FOR LARGE SCALE STATISTICAL INVERSE PROBLEMS

A Dissertation

by

KAINAN WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Wolfgang Bangerth
Committee Members,	Jean-Luc Guermond
	Yalchin Efendiev
	Helmut Katzgraber
Head of Department,	Emil Straube

May 2014

Major Subject: Mathematics

ABSTRACT

The Bayesian method has proven to be a powerful way of modeling inverse problems. The solution to Bayesian inverse problems is the posterior distribution of estimated parameters which can provide not only estimates for the inferred parameters but also the uncertainty of these estimations. Markov chain Monte Carlo (MCMC) is a useful technique to sample the posterior distribution and information can be extracted from the sampled ensemble. However, MCMC is very expensive to compute, especially in inverse problems where the underlying forward problems involve solving differential equations. Even worse, MCMC is difficult to parallelize due to its sequential nature—that is, under the current framework, we can barely accelerate MCMC with parallel computing.

We develop a new framework of parallel MCMC algorithms—the Markov chain preconditioned Monte Carlo (MCPMC) method—for sampling Bayesian inverse problems. With the help of a fast auxiliary MCMC chain running on computationally cheaper approximate models, which serves as a stochastic preconditioner to the target distribution, the sampler randomly selects candidates from the preconditioning chain for further processing on the accurate model. As this accurate model processing can be executed in parallel, the algorithm is suitable for parallel systems. We implement it using a modified master-slave architecture, analyze its potential to accelerate sampling and apply it to three examples. A two dimensional Gaussian mixture example shows that the new sampler can bring statistical efficiency in addition to increasing sampling speed. Through a 2D inverse problem with an elliptic equation as the forward model, we demonstrate the use of an enhanced error model to build the preconditioner. With a 3D optical tomography problem we use

adaptive finite element methods to build the approximate model. In both examples, the MCPMC successfully samples the posterior distributions with multiple processors, demonstrating efficient speedups comparing to traditional MCMC algorithms. In addition, the 3D optical tomography example shows the feasibility of applying MCPMC towards real world, large scale, statistical inverse problems.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor professor Wolfgang Bangerth who has been guiding me in every aspect of research. He has also impacted me in my attitudes towards work, life and family.

I would like to express my thanks to professor Jean-Luc Guermond, professor Yalchin Efendiev and professor Helmut Katzgraber for kindly serving on my committee. I thank them for the directions and comments that help shape my doctoral research.

I am grateful to the whole Bangerth research group. Especially, I would like to thank Dr. Timo Heister and Dr. Bruno Turcksin for their help on understanding parallel computing and programming in general.

Last but not least, I am thankful to my family. The support from my parents and my parents-in-law allows me to concentrate on this research. I want to thank my wife Wei who has not much to do with mathematics but with every other aspect of my life. It is her understanding and unconditional support that make this dissertation come into existence. I owe everything to her.

Part of the work presented herein was supported by the National Science Foundation through Award No. OCI-1148116; and by Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGEMENTS iv

TABLE OF CONTENTS v

LIST OF FIGURES vii

LIST OF TABLES x

1. INTRODUCTION 1

1.1 Inverse problems 1
1.2 Deterministic methods 2
1.3 Bayesian inverse problems 6
1.4 Introduction to parallel computing 9

2. SOME EXISTING SAMPLING METHODS 12

2.1 Importance sampling methods 13
2.2 Markov chain Monte Carlo methods 15
2.2.1 Markov chain and its basic properties 16
2.2.2 Metropolis-Hastings algorithm 18
2.2.3 Adaptive proposal for Metropolis-Hastings 22
2.2.4 Diagnostic tests for MCMC algorithms 24
2.3 Existing parallel MCMC algorithms 29
2.3.1 Prefetching MCMC algorithm 30
2.3.2 Parallel tempering method 32

3. THE MARKOV CHAIN PRECONDITIONED MONTE CARLO METHOD 37

3.1 The MCPMC algorithm 37
3.2 Implementing MCPMC for multiple processors 42
3.2.1 Discussion on the parallel efficiency 43
3.3 Example: A two dimensional Gaussian distribution 47
3.4 Example: A multi-modal Gaussian mixture distribution 50

4. APPLICATION: AN INVERSE PROBLEM WITH ELLIPTIC EQUATIONS 56

4.1 Problem description 56
4.2 Enhanced error model 62
4.3 Numerical results 65

5. APPLICATION: THREE DIMENSIONAL OPTICAL TOMOGRAPHY . 71

5.1 The forward model 73
5.1.1 The radiative transfer equation and the diffusion approximation 73
5.1.2 Fluorescence enhanced optical tomography 77
5.2 The Bayesian model of the inverse problem 78
5.3 Deterministic inversion and adaptive mesh refinement 79
5.3.1 Deterministic inversion 80
5.3.2 Adaptive mesh refinement 84
5.4 Stochastic inversion using MCPMC 84

6. CONCLUSIONS 96

REFERENCES 99

APPENDIX A. REVERSIBLE JUMP MARKOV CHAIN MONTE CARLO METHOD 106

LIST OF FIGURES

FIGURE Page

1.1 A graphic illustration of a typical distributed memory cluster. 11

2.1 Auto-correlation function for one of the 64 estimated parameters. Judging from this plot, we may need to use a *"thinning"* of 10,000 samples, that is, picking every 10,000th sample in the sequence to compute any estimate. 21

2.2 An illustration of the burn-in period during sampling. In this example, the true value of the parameter is 1. It is advised to discard the first 10^4 samples to have a more accurate mean estimator. But this means we have to abandon 1/3 of the total computing effort. 21

2.3 One single iteration of the prefetching MCMC with 7 processors. . . . 31

2.4 Theoretical speedups and efficiencies of the prefetching algorithm for different number of processors. Since in the inverse problems the computing time dominates the communication time, these theoretical estimates should be close to the data from numerical experiments. . . . 33

2.5 A sequence of (unnormalized) tempered density functions for a mixed Gaussian distributions. 35

3.1 A graphical illustration of the modified master-slave architecture with which we implement the MCPMC algorithm. 44

3.2 An illustration of the two dimensional Gaussian distribution problem where we try to sample the distribution centered at $(5, 0)$ using a MCPMC with the preconditioning chain being an MCMC sampling the "approximate" distribution centered at $(0, 0)$. 48

3.3 Convergence comparison of samplers with different perturbation steps. The x-axis is the total number of perturbation evaluations taken on the accurate model. So, for the sampler with $P = 10$, the total number of samples shown in this plot is 10^7 because obtaining each sample requires running a short chain of 10 steps. On the contrary, the sampler with $P = 1$ has 10^8 samples shown in the plot. Even so, samplers with more perturbations demonstrate a better convergence. 51

3.4 Sampling the two dimensional Gaussian mixture distribution: (a) shows the centers of the ten modes; (b) shows the result of 1000 samples from an MH sampler; (c) shows the result from the MCPMC with 6 chains and 1000 samples on the sixth chain; (d) shows the sample paths for the same samples as in (c). 54

3.5 1000 samples from the first chain to the sixth chain of the MCPMC algorithm. The temperature decreases with respect to $i, i = 1, 2, \cdots, 6$. The i-th chain serves as a preconditioner for the $i+1$-th chain which fetches samples from the preconditioner and filter samples according to its own probability distribution. 55

4.1 Generation of the synthetic data. The top left figure shows the reference coefficient field $x(z)$, the top right figure shows the solution $p(z)$ using the reference coefficient and the bottom figure shows the adaptively refined mesh for obtaining the solution. 57

4.2 Four samples from a total variation prior distribution. Pixel values vary across the range of parameters and the pictures are blocky. . . . 59

4.3 Four samples from a Gaussian prior distribution. Pixel values are independent of the values of their neighbor. 60

4.4 The meshes used for the approximate model (left) and the accurate model (right). 62

4.5 Error comparison for a plain MCMC algorithm with either a coarse grid model or an EEM using one million samples. The x-axis is the 64 pixels, i.e., the components x_i of the parameter which appears as a coefficient in the elliptic equation. The y-axis is the error $|\bar{x}_i - x_i^*|^2$, i.e., the square error between the mean estimator and the real value. . 65

4.6 Using the same data as above, the figure shows a comparison of the histogram approximated density functions for the marginal distribution of three coordinates: x_5, x_{34} and x_{51} between the coarse grid, the fine grid and the enhanced error models. 66

4.7 Left: Conditional mean reconstruction using 10^6 samples from the MH sampler on the fine grid. Right: Conditional mean reconstruction using 10^6 samples from the MCPMC chain that evaluates on the accurate model. 69

4.8 Sampling error as defined in (4.3) for the coarse level MCMC (green), fine level MCMC (blue) and MCPMC (red). 69

4.9 Sampling error plotted as a function of estimated running time for the coarse level MCMC (green), fine level MCMC (blue) and MCPMC (red). 70

5.1 Left: Mesh that is used for generating the synthetic data. Right: Diffractive excitation light that illuminates the tissue. 86

5.2 Meshes used in the MCPMC sampling: the top left figure shows the parameter field discretization, the top right figure shows the coarse grid for state variables and the bottom figure shows the fine grid for state variables. 87

5.3 A cross section of the object showing the conditional mean (CM) estimates with coarse chain MH, fine chain MH and MCPMC sampling, also in comparison with the MAP estimate computed from the deterministic inversion. 89

5.4 Comparison of the range of cells whose standard deviation is above a certain threshold (0.00125). Left: Standard deviation using samples from the MH sampler on the fine grid. Right: Standard deviation using samples from the MCPMC sampler. 90

5.5 Comparison of histograms at pixels which have estimated magnitude greater than 60% of the maximal magnitude—these are considered as the target pixels. The histogram from MCPMC is plotted in blue and that from the fine MH is plotted in red. 92

5.6 Autocorrelation function of randomly selected parameter components plotted for autocorrelation time up to 3000. In each plot, the red curve is for the fine MH sampler, green for the coarse MH sampler and blue for the MCPMC sampler. 95

A.1 Upscaled results for a sample well log: the left two plots are the histogram of the boundary locations and the estimated upscaling with a fixed number of layers; the right two plots are the histogram of the boundary locations and the estimated upscaling with variable number of layers and reversible jump MCMC. In the upscaled comparison plots, the wiggly solid line is the true well log, and the smoother lines are the mean, 10% and 90% upscaled estimates as indicated in the plot. Plot obtained from [21]. 110

LIST OF TABLES

TABLE Page

3.1 Average acceptance ratios (AR) in the update stage when using different numbers of chains. The "center distance" column shows the distance between mean vectors of neighboring distributions. 49

5.1 Comparison of conditional mean (CM) and standard deviation (stddev) estimation between MCPMC and fine MH samplers at pixels which have estimated magnitude greater than 60% of the maximal magnitude—these are considered as the target pixels. We also compute a relative error defined as $(\text{mean}_{\text{MCPMC}} - \text{mean}_{\text{fine-MH}})$ $/\text{stddev}_{\text{fine-MH}}$, i.e., we compute the difference between the mean estimates from both samplers normalized by the standard deviation of the fine MH sampler. 93

5.2 The integrated autocorrelation time (IACT) and the mean square jump (MSJ) computed for several components of the parameter vector. In the table, fine MH is abbreviated as "f" and MCPMC is abbreviated as "m". 94

1. INTRODUCTION

1.1 Inverse problems

Much of the motivation in applied mathematics stems from the desire to understand the behavior of systems or to make predictions in different situations. We derive mathematical models—mostly in the form of equations—to achieve this purpose. A mathematical model is usually a general description that arises from our understanding of a system. For example, Hooke's law expresses that the force that stretches or compresses a spring to a certain distance is proportional to that distance. However, for such a model to be of use, we have to understand quantitatively the "parameters" that are underlying each system. In the example of springs, different springs could have different ratios of proportion between the force and the displacement and it is imperative to "measure" that ratio so that we can use Hooke's law to predict the forces. In general, we use the term "parameter" to indicate any system configuration that affects the system's behavior. Some usual examples of parameters include equation coefficients (as in the previous example), source terms or boundary conditions.

The mathematical field of inverse problems studies mathematical techniques to infer such parameters in a model. One of the goals for the inference is to estimate the parameters so that predictions produced by such a configured model coincide with data from physical measurements. With these parameters we will be able to predict the data in the future by simulating the configured model. Another goal is to quantify uncertainties of the inference. It has been well understood that uncertainties lie in every place of the modeling process: the measurements could have errors, the mathematical model could have been simplified to suit computational needs, and the

computer codes may not be completely precise due to the limitations in floating point precision, storage or computing time. Therefore, we have to address the question of how much risk there is to use the configured model.

Inverse problems have wide applications in biomedical imaging and geophysical exploration methods where direct examination of the body interior is infeasible. For example, optical tomography methods [1, 8] use light pattern observations through a body tissue to infer absorption and scattering coefficients in a mathematical model that describes photon movements. These coefficients in turn express the variations inside the tissue and can help detect, for example, the existence of a tumor. Another example are subsurface modeling/characterization of oil reservoirs and aquifers. In these problems, people are often interested in estimating the permeability field—a measure of the ability of a porous medium to allow fluids to pass through it. Since the permeability varies throughout the entire reservoir which is underneath the earth surface, it is impossible to measure it directly [54]. In this context, static data such as core/well logs and dynamic data such as water cut at producing wells are used to infer the permeability. A flow equation, which is derived from mass conservation and Darcy's law, bridges data with the unknown permeability: the unknown permeability is a coefficient in the flow equation and the state variables, i.e., the pressure and the saturation, relate to the observed data.

1.2 Deterministic methods

Deterministic methods solve inverse problems by searching for a reasonable parameter that can produce the model output that matches physical measurements. There are a variety of ways to formulate and solve deterministic inverse problems, among which we briefly describe the direct inversion, the singular value decomposition (SVD) and the Tikhonov regularization methods.

Direct inversion methods solve for system parameters through reversing the computation of the forward model. One such example is the layer stripping method in geophysical inversion [58]. In a horizontally layered media, the plane wave response on the top can be expressed as an accumulated effect: at each layer, the global reflection response can be computed from the physical property of the current layer in combination of the response computed for a lower layer, and by repeating this process from bottom up we can compute the response on the top surface—some datum we are able to observe. This consists of the forward model. In the inverse problem, starting from the observed data from top layer, we can remove the effect of each layer to get the response of the next layer, which, together with some prior information, can be used to deduce the acoustic parameters that correspond to the physical properties of each layer.

In most inverse problems, the recursive feature that allows for direct inversion does not exist, and hence we need to consider more general ways of conducting inversion. Suppose $f : H_1 \to H_2$ is a compact parameter-to-observation operator on two Hilbert spaces, the goal of inverse problems can be stated as solving the following equation for the system parameter x given some observed data y

$$f(x) = y. \tag{1.1}$$

This problem may not be well posed, for example, in cases where the dimension of H_1 is larger than that of H_2. One way to mitigate this ill-posedness is to make use of the singular value system (λ_n, v_n, u_n) of the operator f. It is well known that $\{v_n\}_n$ and $\{u_n\}_n$ are orthogonal sets in H_1 and H_2 respectively, the singular values

λ_n decrease to zero and we can always decompose the operator as

$$f(x) = \sum_n \lambda_n(x, v_n)u_n.$$

With additional regularity on the data y and the singular system, we may express the solution as

$$x = x_0 + \sum_n \frac{1}{\lambda_n}(y, u_n)v_n, \qquad \forall x_0 \in \mathrm{Ker}(f).$$

Though appearing reasonable, the above expression does not make much practical sense because neither is the solution unique unless $f(x)$ is injective, nor can we in general compute the infinite sum on the right hand side. Even worse, there is no guarantee that the observed data $y \in H_2$ falls in the range of the operator f in which cases no exact solution exists. A work-around to these difficulties is to solve instead of the original equation (1.1), a truncated version

$$f(x) = P_k y, \tag{1.2}$$

where $P_k : H_2 \to \mathrm{span}(u_1, u_2, \cdots, u_k)$ is a projection. If we further restrict the solution to be orthogonal to $\mathrm{Ker}(f)$, then the solution x_k is guaranteed to uniquely exist, and it has a nice expression as

$$x_k = \sum_{n=1}^{k} \frac{1}{\lambda_n}(y, u_n)u_n.$$

See [39] for more discussion on this truncated singular value decomposition (TSVD) way of inversion.

An immediate question for the TSVD would be how to choose the number k of truncated terms. This is generally difficult to determine. When the singular values

decay rapidly, the solution x_k would blow up easily. Thus, in order to obtain a more stable solution, we resort to regularization methods. Formally, the regularization methods frame the inverse problem as a least squares problem

$$x^* = \arg\min_x ||y - f(x)||^2,$$

namely, instead of an exact inversion, we search a solution that minimizes the difference between the model prediction and the observation. To impose some control to the solution x^*, additional knowledge of parameters in the formulation is imposed. Some examples of such information are the experts' guess of actual parameter values, the smoothness of the parameters or how the parameter at one point should connect to its neighbors. In deterministic inverse problems, this additionally imposed knowledge is called regularization.

Tikhonov regularization is a classical regularization method [62]. It is formulated as minimizing the following expression

$$F_\delta(x) = ||y - f(x)||^2 + \delta||x||^2.$$

This formula expresses the idea of controlling the misfit between the predicted and measured data while controlling the norm of the parameter itself. Therefore, we can hope to avoid blowing up situations that are likely to be encountered in TSVD. The use of the additional regularization term guarantees that a unique optimal solution can be obtained, as is stated by the following proposition:

Proposition 1.2.1. *Let f: $H_1 \rightarrow H_2$ be a compact operator with the singular system (λ_n, v_n, u_n). Then the Tikhonov regularized solution exists, is unique, and is given*

by the formula

$$x_\delta = (f^* f + \delta I)^{-1} f^* y = \sum_n \frac{\lambda_n}{\lambda_n^2 + \delta} \langle y, u_n \rangle v_n,$$

where f^* is the adjoint operator of f.

See [39] for a proof of this proposition.

1.3 Bayesian inverse problems

Bayesian inverse problems belong to the category of statistical inverse problem methods where all the model variables are considered as random variables. Therefore, instead of targeting at a single best estimate x^*, statistical inverse problems aim at estimating a probability distribution from which we can extract information. Formally, suppose $x \in \mathbb{R}^l$ is the parameter to estimate and $y \in \mathbb{R}^m$ are the measurements, then the solution to a Bayesian inverse problem is a conditional distribution $\pi(x|y)$. By applying the classical Bayes' rule, this *posterior distribution* can be interpreted as follows:

$$\pi(x|y) = \frac{\pi(y|x)\pi(x)}{\pi(y)}, \tag{1.3}$$

where $\pi(y|x)$ is the *likelihood* between the predictions from any possible parameter x and the real observation y, $\pi(x)$ is the *prior* knowledge such as the support, the smoothness and the likely values for the parameters. The normalizing constant $\pi(y)$ is not straightforward to estimate. Fortunately though, it needs not be computed in the algorithms we are interested in.

The Bayesian way of modeling inverse problems has advantages in the following aspects. For single value estimation, it provides us with various estimators. One is

the *maximal likelihood* (ML) estimator

$$x_{\mathrm{ML}} = \arg\max_x \pi(y|x), \tag{1.4}$$

another is the *maximal a posteriori* (MAP) estimator

$$x_{\mathrm{MAP}} = \arg\max_x \pi(x|y), \tag{1.5}$$

and the third estimator, the *conditional mean* (CM) estimator, involves an integral

$$x_{\mathrm{CM}} = \int_{\mathbb{R}^n} x\pi(x|y)dx. \tag{1.6}$$

In addition, it provides us with estimators that can be used for estimating uncertainties and the correlation between parameters. The *conditional covariance* estimator

$$\mathrm{cov}(x|y) = \int (x - x_{\mathrm{CM}})(x - x_{\mathrm{CM}})^T \pi(x|y)dx \tag{1.7}$$

is such an estimator.

If the prior and likelihood distributions happen to be Gaussian, the first two estimators are identical to the un-regularized and regularized deterministic methods, respectively. Many existing optimization algorithms [53] can be used to find these two estimators, most of which require computing derivatives of the operator. The third estimator uses expectation of the distribution which requires computing an integral instead of computing derivatives. The fourth estimator estimates the variance of each component as well as correlations between components. The spread and correlation information provided by this estimator is of importance in applications because it quantifies uncertainty of our inference, which is non-existent in

deterministic inversion methods. In this sense, Bayesian inverse problems are more versatile estimation methods that provide a more comprehensive view of the parameters. Moreover, the regularization seen earlier in deterministic methods is more or less arbitrary, whereas in Bayesian inverse problems it naturally appears as the prior distribution $\pi(x)$. See [39,61] for introductions on statistical inverse problems as well as comparisons to deterministic methods.

Bayesian inverse problems also ensure greater flexibility on building models. Different distributions can be adopted for both the prior and likelihood distributions. We can express the belief in the smoothness of the parameter field with a Gaussian distribution for derivatives of the parameter, while the faith of a larger chance of having outlier data can be expressed with a heavily-tailed student-t distribution [37]. In a Bayesian setting, it is also possible to model the regularization parameter δ as a hyper-parameter (see, e.g., [11, 63]) so that the weight of importance between the data misfit and parameter control is itself subject to uncertainty. The hyper-parameter is updated during the course of estimation automatically so that it is not affected by subjectivity.

This framework also permits separation of uncertainties. It is feasible to split physical experiment error, modeling inaccuracy and numerical discretization error with distinct distributions in the Bayesian methods [40, 41], which does not have a counterpart in the deterministic inversion where the regularization term serves merely to retain a mathematical well-posedness rather than depicting uncertainties. For example, an extra prior estimation can be made specifically for the discretization error, and hence producing a more precise model. We will demonstrate such a modeling example in Section 4.2.

It is worth pointing out that although we describe the settings of Bayesian inverse problems under finite dimensional assumption, the Bayesian framework can

be extended to infinite dimensional function spaces where densities with respect to Lebesgue measure are not available. The recent survey [60] gives a theoretical discussion regarding this topic.

1.4 Introduction to parallel computing

Since this thesis is mainly about building new algorithms applicable to parallel computing rather than seeking the most efficient implementations, we only briefly introduce some basics to parallel computing here. For a thorough introduction in aspects of different architectures, main programming paradigms, classical algorithmic applications, etc., see, for example, [30].

In general, parallel computing is any type of computing practice that uses a group of processors to finish the task. It has become an important part of computing from three reasons. First, to capture more features in a scientific model, a higher accuracy is usually desired. But higher accuracy demands more complex models, and hence more computing power to solve within a reasonable amount of time. Many problems in geoscience, atmospheric science and nuclear engineering are so complicated that they can only be computed with thousands or even tens of thousands of processors running together. Secondly, more and more researchers now can have access to powerful machines that, for example, are in petascale—they can reach over one petaflops (i.e., 10^{15} floating point operations per second) at peak time. Furthermore, the traditional trend of increasing computing power by increasing frequency in single CPUs has ceased due to the fact that higher frequency computing units consume too much energy. On personal computers, these days even on portable devices, multiple cores have become standard. Therefore, studying and developing new algorithms that are suitable to run in parallel has an increasing necessity.

There are different models that the member processes in a parallel system com-

municate with each other. Two prevailing models are the *shared memory* model and the *distributed memory* model. In a shared memory model, all processes can access the same memory simultaneously. This provides convenience in parallelising read only instructions because there is no difference between the sequential and parallel version of such instructions. However, for instructions that write into the shared memory, it is possible that multiple processes write into the same address simultaneously, or the order of different processes writing to the memory can change the outcome. These possibilities pose potential bugs for executing programs, and one common way to address such problems is to adopt *mutual exclusion*. Threading (e.g., POSIX) and directives (e.g., OpenMP) are popular programming paradigms on shared memory systems.

In a distributed memory system, each processor can only work on local data, and explicit communications have to happen whenever they need to access remote data. This is a common architecture on modern computing clusters. A typical structure of a distributed memory system consists of multiple computing nodes (see Figure 1.1), where each node has several processors/cores and a memory that is shared by the processing units on the same node, i.e., each node can be viewed as a shared memory system. The communication between different nodes is achieved through a network such as ethernet or infiniband and via *message passing* paradigms. In contrast to the shared memory system, it is important for each process on a distributed memory system to know where the data are. The fundamentals of message passing paradigms are send() and receive() functions as well as the identity of different processes. Message Passing Interface (MPI) (see, e.g., [33, 59]) is a standardized and portable message-passing system for writing programs for distributed memory systems.

One major concern in designing message passing algorithms is the performance overhead given by communication between processes. This is especially important

when the number of available processes is huge, or if the amount of computing work allocated to each process is relatively small. In all of our problems, we always neglect the time for a message to be passed between processes. This assumption is valid as long as the computation time on each process dominates, which is the case for the problems considered in this thesis where the tasks on each processor involve solving partial differential equations. However, it should be guaranteed that the message can be sent whenever it is needed. We design and implement our new algorithm following this idea, see Section 3.2 for a detailed discussion.

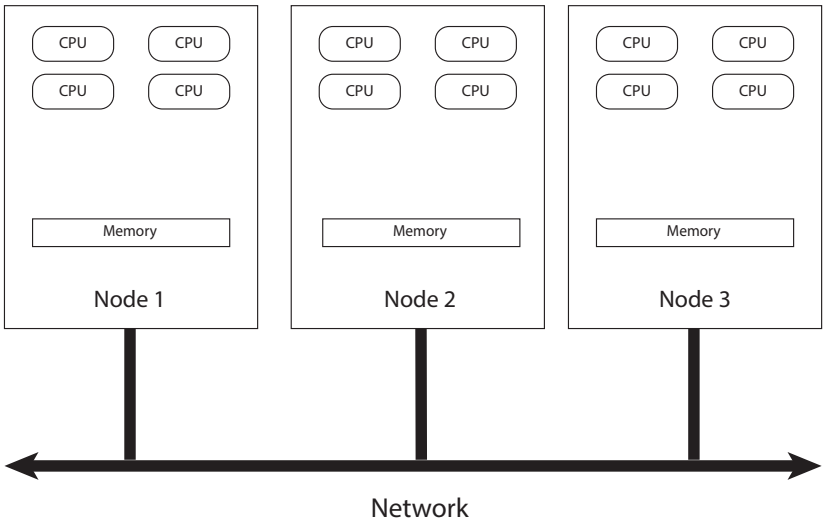


Figure 1.1: A graphic illustration of a typical distributed memory cluster.

2. SOME EXISTING SAMPLING METHODS

To compute the conditional mean estimator and the conditional covariance estimators, it is always necessary to calculate integrals

$$\mathbb{E}_\pi(h(x)) = \int h(x) d\pi(x) \tag{2.1}$$

over the parameter space. In cases where the forward model is complex and the parameter space is of high dimension, computing such integrals analytically becomes infeasible. Traditional quadrature rules also cannot apply because the number of quadrature points grows with dimension. Monte Carlo methods address this issue. The idea of Monte Carlo is to evaluate the target distribution by repeatedly generating random samples from this distribution. Aggregated results are then computed using the ensemble of samples as a discrete approximation of the underlying distribution.

The conventional Monte Carlo method draws identically independent (i.i.d.) samples directly from a probability distribution, the central limit theorem guarantees the convergence of the sample expectation value to the real mean value of the sampled distribution at a rate of $O(n^{-1/2})$, where n is the number of samples drawn. However, it is usually infeasible for inverse problems because, in most cases, it is impossible to derive an analytical form for the posterior distribution with which one can draw independent samples. Therefore, advanced random sampling techniques are developed to circumvent the need of sampling directly from the posterior distribution.

In the following sections, we will briefly describe two classes of such algorithms. The *importance sampling* uses auxiliary distributions that are easier to sample from

to produce independent candidates and then computes a weight for each of these samples. The generated samples, together with their computed weights, is considered a quadrature for any integration under this probability distribution. The *Markov chain Monte Carlo* (MCMC) uses correlated samples to traverse the sample space and adopts a rejection scheme so that samples of higher probability can be repeated in the ensemble. While the importance sampling method is apparently a parallel algorithm by its nature, it usually performs poorly in practice, especially for complex cases, due to its inability to adjust itself towards regions of interest. On the other hand, MCMC type algorithms are shown in applications to provide the capability of finding the high probability region gradually, but it has to run sequentially to satisfy its Markov chain property.

We will describe both algorithms, discuss the causes for their advantages and disadvantages and some practical ways to measure algorithm efficiency. It is these observations that lead to the idea of our new algorithm in the next chapter, which can be viewed as a hybrid of importance sampling and MCMC, ensuring sampling stability while allowing parallel sampling. In addition, we list some existing parallel MCMC algorithms in Section 2.3.

2.1 Importance sampling methods

The central idea behind importance sampling for a target distribution $\pi(x)$ is to use an auxiliary probability distribution $g(x)$ that resembles the target while being easy to sample from. Once such a $g(x)$ is constructed, we can rely on the simple change of measure formula for approximation: for any L_1 function $h(x)$, we will have

$$\mathbb{E}_\pi(h(x)) = \int h(x)\pi(x)dx = \int h(x)\frac{\pi(x)}{g(x)}g(x)dx.$$

In practice, we can continuously generate i.i.d. samples $x_1, x_2, \ldots, x_n$ from $g(x)$

and compute the importance weight

$$\omega(x_i) = \frac{\pi(x_i)}{g(x_i)}. \tag{2.2}$$

Then, we can build a quadrature with the pairs $\{x_i, \omega(x_i)\}$: to estimate the integral of any function $h(x)$ under $\pi(x)$, we can use

$$\mathbb{E}_\pi(h(x)) \approx \frac{\sum_{i=1}^n \omega(x_i)h(x_i)}{\sum_{i=1}^n \omega(x_i)},$$

where the appearance of the sum in the denominator is justified by the fact that $\omega(x_i)$'s are usually computed up to a normalization constant, and the effect of this missing constant can be cancelled by also dividing by the sum of ω's.

It is difficult to accurately measure the efficiency of the importance sampling algorithms. A "rule of thumb" is to use the *effective sample size* (ESS) to measure the discrepancy between the trial distribution and the target distribution. Suppose a total of n samples is generated using importance sampling, the ESS is defined as

$$\mathrm{ESS}(n) = \frac{n}{1 + \mathrm{var}_g[\omega(x)]},$$

where var_g is the variance of the importance weights. In practice, the variance is estimated through the coefficient of variation defined by

$$\mathrm{var}_g[\omega(x)] \approx \frac{\sum_{i=1}^n (\omega_i - \overline{\omega})^2}{(n-1)\overline{\omega}^2},$$

where $\overline{\omega}$ is the sample mean of the weights.

The ESS can be interpreted as the number of samples from i.i.d. draws that can achieve the same accuracy as n importance samples. If the shape of the trial and

target distributions are very different, the variance of the weights can be huge, and thus the ESS can be small, which in turn indicates a poor efficiency in the constructed sampler.

This also reveals the fact that the key to a successful importance sampling lies in the construction of the trial distribution. It has to meet two requirements, namely, it has to be close to the target distribution and we should be able to generate independent samples distributed according to $g(x)$ directly. Both requirements pose challenges to finding satisfying trial distributions for the posterior distributions in Bayesian inverse problems where nonlinearity and high dimensionality are the main features. Therefore, though appearing as a simple and powerful sampling method, importance sampling is rarely applied in Bayesian inverse problems directly. However, its idea of computing the importance weights is borrowed by advanced algorithms such as the sequential Monte Carlo [22, 23]. Our new algorithm in the next chapter is also partially inspired by this idea.

2.2 Markov chain Monte Carlo methods

Markov chain theory emerges in the Monte Carlo literature in 1953 by Metropolis [51] to stabilize the parameter space exploration using a random walk dependent on the current state. In 1970, Hastings [36] extended the range of proposal functions from random walk to more general functions. See [55] for a historical summary. Though evidently powerful, Markov Chain Monte Carlo (MCMC) methods did not prevail until the 1990s, mainly due to the lack of efficient computing technology. The visionary work [27] opened doors for MCMC to solving a wide range of problems from biostatistics, physics, geosciences and finance.

2.2.1 Markov chain and its basic properties

To state the fundamental properties of MCMC algorithms, let us first briefly introduce several definitions about Markov chains. A *Markov chain* is a discrete stochastic process $\{X_j\}$ that satisfies the following property:

$$\pi_{X_{j+1}}(B|x_j, \ldots, x_0) = \pi_{X_{j+1}}(B|x_j),$$

where X_j is the j-th random variable in the (discrete) process, x_j is a realization of the random variable X_j, B is any measurable set and π_X is the probability distribution defined by the random variable X. Simply speaking, any state of the random variable X_{j+1} in the process depends on the history of the process only through that of its direct predecessor X_j. A transition kernel $P : \mathbb{R}^n \times \mathcal{B} \to (0, 1)$ defines the probability to move a point x to a Borel set B, and $P(x, B)$ defines a *time homogeneous* Markov chain $\{X_j\}$ with the Markov property defined by

$$\pi_{X_{j+1}}(B_{j+1}|x_j) = P(x_j, B_{j+1}).$$

It is called time homogeneous because the transition kernel stays the same for any time index j. In fact, the transition kernel defines the entire process recursively for each of the random variables X_j, $j = 1, 2, \ldots$ in the process, that is:

$$\pi_{X_{j+1}}(B) = \pi_{X_j} P(B) := \int_{\mathbb{R}^n} \pi_{X_j}(x) P(x, B) dx.$$

So in contexts we may freely use P to represent the Markov chain it defines. For a given Markov chain with a transition kernel P, a probability distribution $\pi(x)$ that

satisfies

$$\pi(B) = \pi P(B) := \int_{\mathbb{R}^n} P(x, B)\pi(x)dx$$

is said to be an *invariant* distribution of P. Later we will demonstrate that a Markov chain with minimal properties will converge to its invariant distribution. This property is the backbone of all MCMC algorithms.

Now we introduce several concepts that characterize Markov transition kernels. A Markov transition kernel P is *irreducible* with respect to a distribution π if for all $x \in \mathbb{R}^n$ and $B \in \mathcal{B}$ with a positive measure under π, there is a $k > 0$ such that $P^{(k)}(x, B) > 0$ where we define:

$$P^{(k)}(x, B) = \int_{\mathbb{R}^n} P(x_{k-1}, B)P^{(k-1)}(x, x_{k-1})dx_{k-1}.$$

Consequently, for any starting point x, and any set B in the support of π, we should be able to use P to move x to B in finitely many steps.

For an irreducible kernel P, we say P is *periodic* if there is a set of disjoint nonempty sets $\{E_1, \ldots, E_m\} \subset \mathbb{R}^n,\ m \geq 2$, such that for all $j = 1, \ldots, m$ and any $x \in E_j$, we have $P(x, E_{(j+1)mod(m)}) = 1$. That is to say, a periodic kernel will generate a Markov chain that stays in a loop constantly. We call a kernel that is not periodic an *aperiodic* transition kernel.

When a Markov chain with transition kernel P satisfies the irreducibility and aperiodicity requirements, which is often true in the designed MCMC algorithms, it has an ergodicity property that is essential to the usability of MCMC:

Proposition 2.2.1. *Let π be a probability measure and $\{X_j\}$ a time homogeneous Markov chain with an irreducible and aperiodic kernel P that has π as an invariant*

measure. Then for $\pi-$almost all $x \in \mathbb{R}^n$,

$$\lim_{N\to\infty} P^{(N)}(x, B) = \pi(B) \text{ for all } B \in \mathbb{B},$$

and for all $f \in L^1(\pi(dx))$,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{j=1}^{N} f(X_j) = \int_{\mathbb{R}^n} f(x)\pi(dx).$$

This theorem suggests that to study a probability distribution $\pi(x)$, we may simulate a realization, denoted as $\{x_1, x_2, \ldots, x_n, \ldots\}$, of a Markov chain P with $\pi(x)$ being its invariant distribution. Then, the probability of any Borel set $B \in \mathcal{B}$ can be approximated by the relative number of samples in that set regardless of the starting point of this chain. Moreover, the expectation value of any function $f(x)$ over the distribution π can be approximated by the *ergodic average* of such a realization, according to the theorem above.

2.2.2 Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is an MCMC algorithm that forms the basis to many advanced methods. At iteration j, a trial sample x^* is produced using a proposal function $q(x_{j-1}, x^*)$ where x_{j-1} is the previous state. The probability distribution $\pi(\cdot)$ is evaluated at x^*. According to an update formula, the Markov chain chooses to either go to the trial sample (i.e. *accept* the trial) or merely replicate the previous state (i.e. *reject* the trial).

Thus, an iteration of Metropolis-Hastings algorithm will look like:

Algorithm 1 Metropolis-Hastings Algorithm

Choose a starting point x_0. For iteration $j = 1, 2, \ldots$

- Propose x^* using $q(x_{j-1}, \cdot)$,
- compute $\pi(x^*)$ and compute $\alpha(x_{j-1}, x^*)$ where α is defined as:

$$\alpha(x, y) = \min\left\{1, \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}\right\},$$

- draw $u \sim \mathbf{U}(0, 1)$,

  If $\alpha(x_{j-1}, x^*) \geq u$

  let $x_j = x^*$,

  Else

  let $x_j = x_{j-1}$.

A commonly used proposal function is the *random walk* distribution which is a multivariate normal distribution $\mathcal{N}(x_{j-1}, \Sigma)$. In this cases, the proposal is symmetric, i.e., $q(x, y) = q(y, x)$, and the update ratio can be further reduced to $\alpha(x, y) = \min\{1, \pi(y)/\pi(x)\}$. For high dimensional problems, the choice for a fixed covariance matrix Σ is an art. [57] studies this and proposes that the optimal choice of Σ should give an acceptance ratio that is close to 0.234.

The Metropolis-Hastings algorithm has been applied in many inverse problems that require Monte Carlo type samplings. It provides reliable performance thanks to the prudent proposal and update steps that link the adjacent samples. However, the algorithm is computationally very expensive. The number of samples needed to estimate the invariant distribution becomes huge when the parameter space is of

high dimension. In the following, we emphasize two features that result in a large sample number in MH type simulations.

Autocorrelation: Due to the locality of the proposal function $q(x, y)$, consecutive samples are likely to correlate strongly with each other and hence the ensemble would only have a representating ability equal to that of a much smaller number of independent samples. One of the measures for this intra-chain correlation is the auto-correlation function $\gamma_j(h)$ for some function $h(x)$ of interest. It is defined by

$$\gamma_j(h) = \frac{1}{(N-j)\sigma^2} \sum_{i=1}^{N} \left(h(x_i) - \overline{h(x)} \right) \left(h(x_{i+j}) - \overline{h(x)} \right), \tag{2.3}$$

where j is called the *lag* of the process, σ^2 is the sample variance and $\overline{h(x)}$ is the mean of $h(x)$ over all samples. One practical strategy is to select every k-th sample from the chain where the auto-correlation function γ_j is small for $j \geq k$. This "thinning" procedure can effectively reduce sample correlations and the statistics computed from this subset of samples can be more accurate. We illustrate such an auto-correlation function of a single parameter in Figure 2.1. It comes from chapter 4 where a 64 dimensional parameter is sampled with MCMC method.

Burn-in: At the beginning of the sampling process which starts from an area away from the high probability region, or when the Markov process has not come to the stationary distribution, it will spend several samples to guide itself towards the target region. In practice, this initial segment is called *burn-in* period and samples in this period are discarded to guarantee a more accurate estimation from the finite number of samples. See Figure 2.2 for an illustration of the burn-in period during sampling.

What adds to the problem of needing many samples in MCMC is that in inverse problems the evaluation of each sample is expensive because the forward model in-

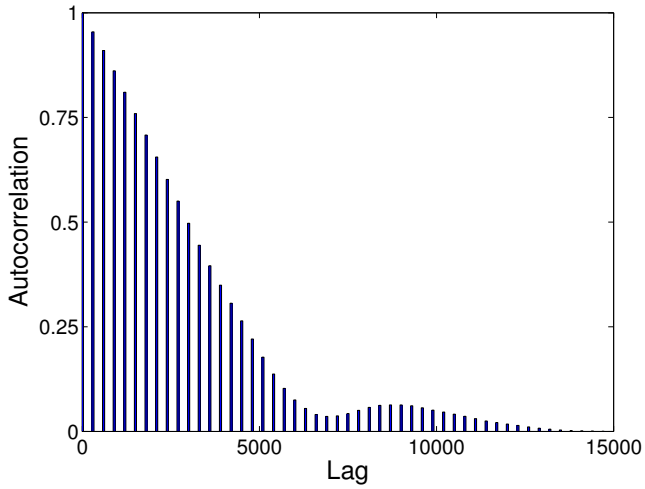


Figure 2.1: Auto-correlation function for one of the 64 estimated parameters. Judging from this plot, we may need to use a *"thinning"* of 10,000 samples, that is, picking every 10,000th sample in the sequence to compute any estimate.

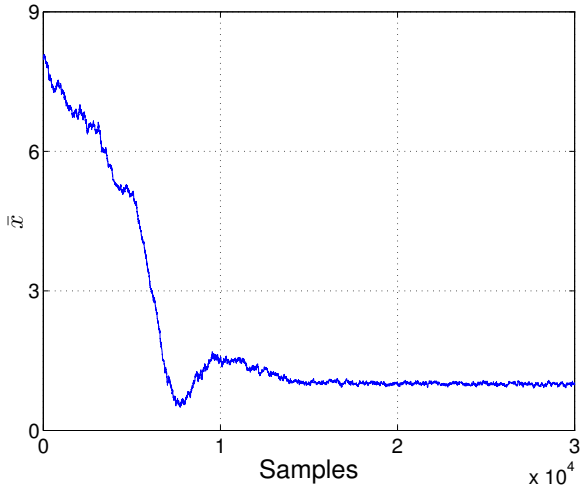


Figure 2.2: An illustration of the burn-in period during sampling. In this example, the true value of the parameter is 1. It is advised to discard the first 10^4 samples to have a more accurate mean estimator. But this means we have to abandon 1/3 of the total computing effort.

volves solving the underlying partial differential equations. For example, in the optimal tomography example in chapter 5, a single evaluation costs about 41.5 seconds, and it takes us about 29 days to collect 60, 000 samples with MCMC. Although parallel computing has become an important means of acceleration and many algorithms have been successfully parallelized, the study of MCMC parallelization is relatively scarce. This is mainly due to the sequentiality of the algorithm where new samples can not be generated until after the accept-reject decision has been made for the current sample. This motivates our research to design new frameworks that enable running MCMC in parallel.

To summarize, in order to make MCMC more tractable for large scale inverse problems, we need to either increase the statistical efficiency of the algorithm or modify the algorithm so that it can be run in parallel. In the following section, we will describe an adaptive MCMC algorithm which could help improve the statistical efficiency of the algorithm. It is worth noting that this technique can be readily incorporated in our new framework introduced in Chapter 3.

2.2.3 Adaptive proposal for Metropolis-Hastings

To ameliorate the slow mixing of random walk Metropolis-Hastings, [35] proposes an adaptive method. In contrast to the plain random walk that always proposes new samples according to the same normal distribution, this method will adapt the proposing distribution using the statistics of past samples. Through this adaptation, we can hope to approximate the shape of the target distribution function around the current sample, thus increasing the chances to land on a state superior than the current one. More specifically, suppose the Markov chain has already generated samples $\{x_1, x_2, \ldots, x_n\}$, the proposal function for the next sample will be a normal

distribution with mean vector

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and covariance matrix Σ_n which is the sample covariance of all n past samples. Since Σ_n is defined as

$$\Sigma_n = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^T - \frac{n}{n-1}\mu_n \mu_n^T,$$

we can update this matrix in a fast recursive way through

$$\Sigma_n = \frac{n-2}{n-1}\Sigma_{n-1} + \mu_{n-1}\mu_{n-1}^T - \frac{1}{n-1}(n\mu_n\mu_n^T - x_n x_n^T).$$

The proposal function is then chosen to be a normal distribution with the covariance matrix:

$$\Sigma_{\text{update}} = s_d(\Sigma_n + \epsilon\mathcal{N}(0,1)I_d),$$

where d is the dimension of the parameter space, s_d is empirically chosen to be $(2.38)^2/d$. The $\epsilon\mathcal{N}(0,1)I_d$ term is a "safety measure" that prevents the update covariance matrix from being singular.

We use the following procedure to compute new proposals which should then give the multivariate normal distribution $\mathcal{N}(\mu_n, \Sigma_{\text{update}})$: at each iteration, we first perform a Cholesky decomposition for the update covariance matrix Σ_{update} to obtain a lower triangular matrix L where L satisfies

$$LL^T = \Sigma_{update},$$

and then we generate a random vector $Z \sim \mathcal{N}(0, I_d)$. The new sample will have the

form $X^* = X_{n-1} + LZ$. In fact, it is straightforward to check that

$$\mathbb{E}(X^*) = \mathbb{E}(X_{n-1} + LZ) = X_{n-1},$$

and that

$$\Sigma(X^*) = \mathbb{E}(LZZ^TL^T) = LL^T = \Sigma_{update}.$$

The adaptive proposal has many applications and is adopted to increase sampling efficiencies in Bayesian inverse problems, for example, in [20]. Together with a delayed-rejection scheme, they constitute the delayed-rejection adaptive Metropolis (DRAM) algorithm [34]. It has been used as a reference algorithm to compare newly developed advanced, single chain MCMC algorithms, for example, in [49]. Theoretically, since the adaptive proposal incorporates information regarding the entire history, rather than the most recent sample, it imposes some difficulty in analysis. It is only until recent years that convergence rates of such adaptive algorithms are estimated. See [4, 56] for a discussion on the theory of the adaptive algorithm.

2.2.4 Diagnostic tests for MCMC algorithms

The convergence of a Monte Carlo method using independent and identically distributed random variables is provided by the classical central limit theorem, saying that the error is bounded by $O(1/\sqrt{n})$, where n is the number of independent samples. On the other hand, it is much more difficult to estimate whether a Markov Chain has converged. Correlation between successive samples in a Markov chain introduces uncertainty, giving difficulty to any attempt of drawing a comprehensive conclusion. To date, there is yet any theoretical support to give a good estimate over this issue in general. However, numerous diagnostic tools exist to test the convergence property of Markov chains in practice. What these examiners do is, rather than assuring that

a chain has already converged, telling that a chain has not reached stationarity. As long as these examiners tell "no" we have to let the chain run longer. In addition, the diagnostic can also verify the speed of mixing of a chain. If a chain is developed in the manner that no matter where we start the chain, it could direct itself to stationarity quickly, the examiner will be able to indicate such a chain as a "good" one. In the paragraphs below, we will briefly review a few diagnostic methods and then detail the convergence standard we prefer in our numerical experiments.

A good overview of such diagnostic tools is [19]. Among the different testing methods, a popular one is the Potential Scale Reduction Factor (PSRF) derived in [28]. Multiple chains are started from different positions that are far away from each other (being "overdispersed"). At convergence, the histograms of each variable from different chains should overlap considerably.

The formal definition of PSRF for a one dimensional variable is as follows. m chains are started from different points, with each generating $2n$ samples. The last n samples are picked, and the within-chain variance W and between chain variance B/n are computed by

$$W = \frac{1}{m(n-1)} \sum_{i=1}^{m} \sum_{j=n+1}^{2n} (x_{ij} - \bar{x}_i)^2$$

and

$$B/n = \frac{1}{m-1} \sum_{i=1}^{m} (\bar{x}_i - \bar{x})^2,$$

where $\bar{x}_i$ is the average of last n samples from chain i and $\bar{x}$ is the average of all $\bar{x}_i$'s.

Taking both the within chain and between chain variances into account we obtain

an estimate of the posterior variance which is defined as

$$\hat{V} = \frac{n-1}{n}W + (1 + \frac{1}{m})\frac{B}{n},$$

and the PSRF factor $\hat{R}$ is defined through dividing this variance by W:

$$\hat{R} = \frac{\hat{V}}{W}.$$

When the chain has converged, the factor will be close to one, which makes it a nice criterion for monitoring convergence. However, this factor is unidimensional, namely, it can merely monitor the behavior of a single variable at once. In order to obtain an integrated view of the convergence of all variables, [15] extended the criterion to multivariate situations. The Multivariate Potential Scale Reduction Factor (MPSRF) is defined similarly to its univariate sibling, with scalar values being replaced by multidimensional vectors, and variances replaced by covariance matrices

$$W = \frac{1}{m(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{ij} - \bar{x}_i)(x_{ij} - \bar{x}_i)^T$$

and

$$B/n = \frac{1}{m-1} \sum_{i=1}^{m} (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T.$$

The definition of $\hat{V}$ as a combination of B and W is as before, whereas the definition of $\hat{R}$ becomes a bit different because now we have to define a proper division for

covariance matrices. [15] defines it as the generalized Rayleigh quotient

$$\begin{aligned}\hat{R} &= \max_a \frac{(\hat{V}a, a)}{(Wa, a)} \\ &= \frac{n-1}{n} + \frac{m+1}{m} \max_a \frac{(B/na, a)}{(Wa, a)} \\ &= \frac{n-1}{n} + \frac{m+1}{m}\lambda_{max},\end{aligned}$$

where λ_{max} is the largest eigenvalue of $W^{-1}B/n$ for positive definite matrices W and B/n.

One feature of the PSRF and MPSRF diagnostics is that multiple chains have to be run from different starting points and it can be time consuming if an online diagnostics is required. Especially, in the context of parallel computing, communication between these multiple chains is demanded to summarize the statistics, which could put an overhead to the algorithms. This is avoided in the alternative estimator—the mean squared error (MSE) estimator [17] that is defined as

$$\begin{aligned}MSE(x) &= \mathbb{E}_x(x - x^*)^2 \\ &= \mathrm{Var}_x x + (\mathbb{E}_x x - x^*)^2,\end{aligned}$$

where $\mathbb{E}_x$ means taking the sample average and x^* is the true parameter. The second equation indicates that the MSE includes both the variance of estimate and the bias of an estimator to its true values.

In problems with real data, the true parameter x^* is in general unknown, and hence it is impossible to directly apply the MSE formula for estimation. In Bayesian inverse problems, even if we generate the synthetic data using a reference parameter, since we also incorporate a prior distribution, the mean value of the posterior distribution may not be the same as the true parameter. Therefore, if MSE were used as

a convergence criterion and the error of a sampling scheme did not converge to zero, it would be difficult to tell whether it stems from a poor converging sampler or from the modeling bias.

To circumvent this dilemma so that we can merely compare the convergence performance of different samplers, we make some efforts to exclude the modeling discrepancy. To achieve this, we may run an MCMC chain for a sufficiently long time which we believe to have converged. We then compute the sample mean x^* and the sample covariance Σ out of this long chain. Then, we are able to compute the error as a weighted mean square error which we call the *Empirical Mean Square Error*:

$$\begin{aligned} EMSE_n(x) &= \mathbb{E}_x(x - x^*)^2_{\Sigma^{-1}} \\ &= \frac{1}{n}\sum_{i=1}^{n}(x_i - x^*)^T\Sigma^{-1}(x_i - x^*). \end{aligned} \tag{2.4}$$

The reference Markov chain is long enough so that we can believe that the x^* is as close to the expected value of the posterior distribution as the best of any other sampler we want to evaluate. Also, the Σ quantifies how much uncertainty there is in the model itself. For example, if the physical experiment is designed such that we are only able to observe data from the surface of an object, it is likely that our model may behave more poorly for inferring parameters deep inside the object which then could indicate a bigger variance for internal parameters in the covariance matrix Σ. It is then legitimate to assume that such uncertainty will live in any sampling method as long as they are sampling the same model. Therefore, by weighing the errors with the inverse of the covariance matrix, we can hope to eliminate some uncertainties from modeling and concentrate on the error brought about by the samplers themselves.

Another useful diagnostic test is to compute the *integrated autocorrelation time* (IACT) [48] which is defined as (again, for some function $h(x)$ of the random vari-

ables)

$$\tau(h) = 1 + 2\sum_{j=1}^{\infty} \gamma_j(h), \tag{2.5}$$

where γ_j is the autocorrelation function defined in (2.3). This number is related to the *effective sample size* (ESS) defined for correlated samples, which, for a total of n samples, has the expression

$$\text{ESS} = \frac{n}{2\tau(h)}. \tag{2.6}$$

The ESS is a description of the number of independent samples which have similar estimated statistics as that of the n (potentially correlated) samples. Therefore, the smaller the IACT is, the larger the ESS is, and hence we may say that the sampler is possessing a better statistical efficiency. In later chapters, we will show how to use EMSE and IACT to measure the performance of different samplers. We will also show that the concept of effective sample size can help us properly quantify the speedup of an MCMC algorithm in parallel computing.

2.3 Existing parallel MCMC algorithms

The most straightforward strategy to use MCMC in parallel is to generate multiple chains starting from points that are far away from each other [28]. Samples are then collected from all the chains to achieve a bigger ensemble of samples than a single chain could achieve within the same amount of wall time. However, there are several practical problems with this setting. First of all, there is no clear criterion to tell whether the starting points have spread far away from each other. It is quite possible that different starting points soon lead the sampler towards a common region of interest and each of the multiple chains only manage to sample this region due to limited number of samples. Samples from different chains are then correlated across chains and the convergence could be slowed. In contrast, a single long chain

might have a better chance of ergodicity. In addition, every chain has to discard the burn-in period whose length is non-uniform (because of the different starting points) and hard to estimate. One strategy is to discard a fixed, but large enough, number of samples for each chain, which means that the number of wasted samples grows proportionally to the number of chains available. All these could reduce the gains from the multiple chain implementation. See [29] for a detailed discussion.

In the following, we will discuss two other parallel MCMC algorithms. They represent two categories of ideas behind the parallelization of MCMC: the single chain parallelization and the multiple chain parallelization. We discuss both the improvements they make and the obvious limitations they possess.

2.3.1 Prefetching MCMC algorithm

A single-chain parallelization strategy called the prefetching MCMC was proposed in [13]. When multiple processors are accessible, it is possible to propose, on the main processor, the candidates for several steps ahead using a computationally inexpensive random walk type distribution. These candidates are then distributed to several processors and can be evaluated simultaneously. Once their probability density function values are computed and sent back to the main processor, the main processor can then perform a series of accept-reject decisions for acquiring multiple samples in the Markov chain. We further illustrate the details of the idea by showing an example of prefetching MCMC that uses a total of 7 processes. In this illustration, we use a binary tree structure to demonstrate the conceived stages: for each node in the tree, the left child stands for the new candidate and the chain will go to left if an "accept" decision is finally made upon it; the right child, on the other hand, stands for the result of the "reject" decision and hence it is an identical sample as its parent.

Example 2.3.1. *At the current sample x_j, we use seven processors to extrapolate three levels at the same time. Firstly, on the main processor, a tree of new samples are spawned using a random walk type proposal function. Each left child of a node is a new candidate while the right child is identical to its parent, as is demonstrated in Figure 2.3. Then, all that needs to be computed is the probability density function value (expensive step) at the seven nodes on level three except for the rightmost one on different processors in parallel. Once these values are gathered, on the main processor, starting from x_j, the root, we perform a sequence of Metropolis Hastings updates and get three new samples x_{j+1}, x_{j+2} and x_{j+3}. This algorithm is robust in the sense that we will always obtain three new samples. We still end up with one single chain that is identical to the original single-processor algorithm. The mixing rate and the burn-in both remain intact.*

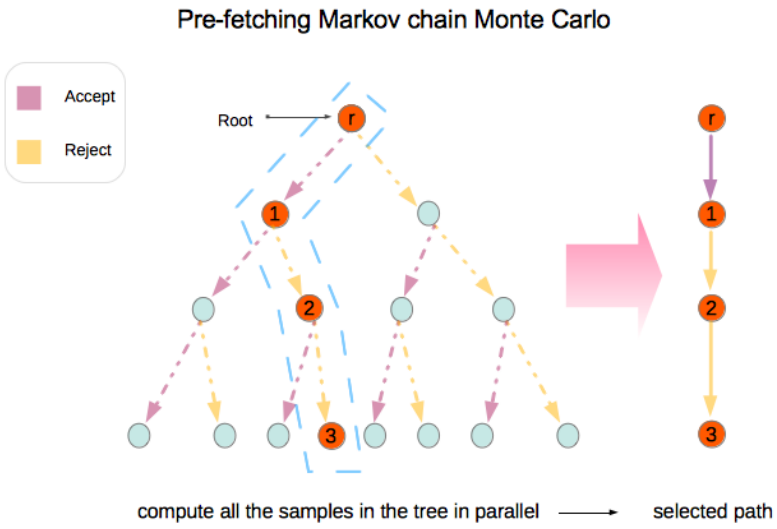


Figure 2.3: One single iteration of the prefetching MCMC with 7 processors.

Prefetching MCMC, from a parallel computing perspective, is not efficient as the

upper limit of its speed up is $\log_2(P+1)$, where P is the number of the processors available. In Figure 2.4 we show the theoretical speed up and the efficiency, the ratio of the speed up over the number of processors available. As can be seen from the plot, the algorithm scales poorly with the number of processors. Besides, the implementation of this algorithm also relies on a fixed, computationally inexpensive proposal distribution such as the random walk, as the candidates are all generated on the main processor and only the evaluation of $\pi(x)$ is parallelized. For advanced MCMC algorithms such as the multiple try Metropolis algorithm [47] where the anchor point for new proposal is undecided until after evaluation is taken place, or the Stochastic Newton's method [49] where each new sample is proposed by solving an optimization problem, the prefetching algorithm cannot be directly applied.

2.3.2 Parallel tempering method

As is discussed above, plainly running multiple chains of MCMC may not be an ideal solution of algorithm parallelization. In this section, we discuss another multiple chain strategy, the *parallel tempering (PT)* algorithm [24], that allows between chain interactions. These interactions empower Markov chains to jump between states rather than being trapped in local optima, and they increase statistical efficiencies in general.

The parallel tempering algorithm deals with a product space $\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_I$, with each $\mathcal{X}_i$ being an identical copy of the original parameter space $\mathcal{X}$. A joint probability distribution

$$\pi(x_1, \ldots, x_I) = \prod_{i \in I} \pi_i(x_i)$$

is assigned to the product space. In implementation, the I chains are run in parallel and the algorithm is defined as:

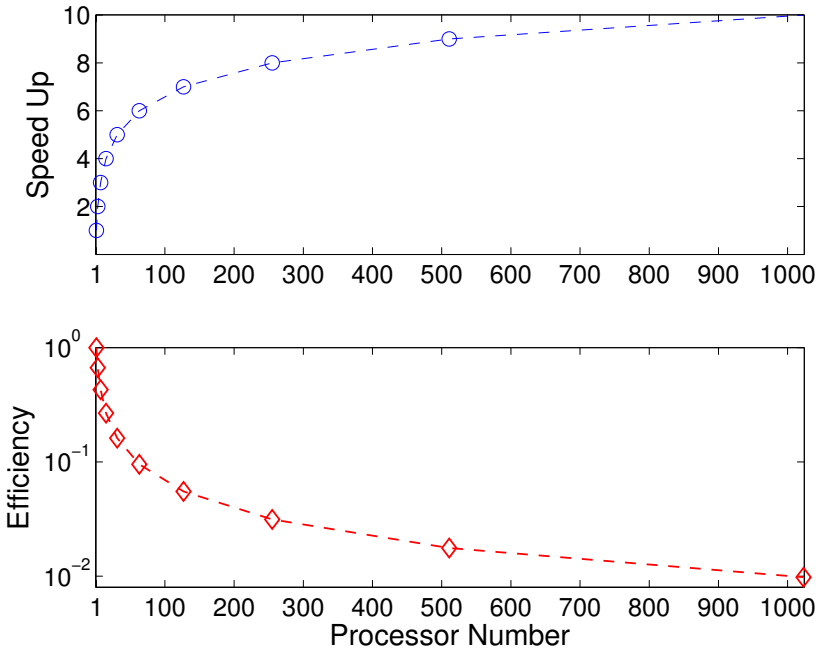


Figure 2.4: Theoretical speedups and efficiencies of the prefetching algorithm for different number of processors. Since in the inverse problems the computing time dominates the communication time, these theoretical estimates should be close to the data from numerical experiments.

Algorithm 2 Parallel Tempering Algorithm

- Let the current state be $\{x_1^{(t)}, \ldots, x_I^{(t)}\}$; draw $u \sim \text{Uniform}[0, 1]$,
- If $u \leq \alpha_0$, we update each $x_i^{(t)}$ to $x_i^{(t+1)}$ by using a Metropolis-Hastings strategy,
- If $u > \alpha_0$, we perform a *swapping step*, i.e., we randomly choose a neighboring pair, say i and $i+1$, and swap $x_i^{(t)}$, $x_{i+1}^{(t)}$ with probability

$$\min\left\{1, \frac{\pi_i(x_{i+1}^{(t)})\pi_{i+1}(x_i^{(t)})}{\pi_i(x_i^{(t)})\pi_{i+1}(x_{i+1}^{(t)})}\right\}.$$

In parallel tempering, the π_i's are chosen as a sequence of "tempered" distributions, namely,

$$\pi_i(x) \propto \exp\{\log \pi(x)/T_i\} = (\pi(x))^{1/T_i}, \tag{2.7}$$

where $1 = T_1 < T_2 < \ldots < T_I$ and π_1 is set to be the original distribution. The above form is similar to the *Boltzmann* distribution in physics where $-\log \pi_1(x)$ serves as the energy and T as the temperature, the setting is hence named. See Figure 2.5 for an example of a sequence of tempered mixed Gaussian distributions. Intuitively, as the temperature increases, the distribution will be more flattened, i.e., the barriers between different regions become smaller, thus allowing the sampler to traverse different areas of the parameter space more easily. When the samples of the original distribution are trapped around a local peak, those sampling the higher temperature distributions may have reached another region of interest. The swap steps in the algorithm permit exchanging information between the states at different energy levels and help the chain on the original distribution to mix faster.

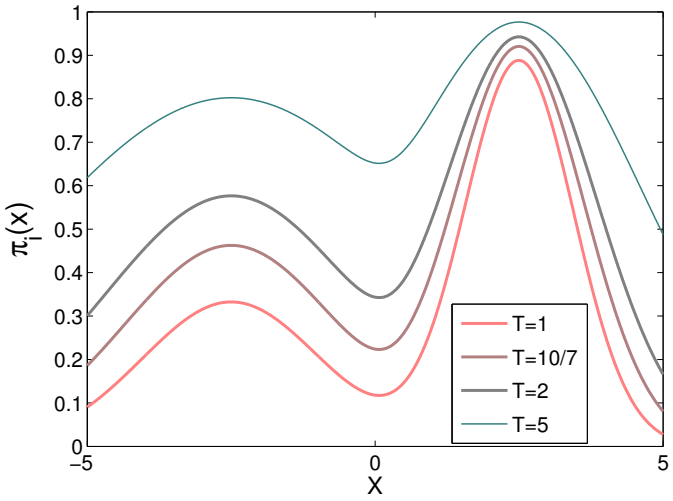


Figure 2.5: A sequence of (unnormalized) tempered density functions for a mixed Gaussian distributions.

A practical issue in implementation lies in the tuning of the temperature ladder. An empirical conclusion is that one should start with choosing the highest temperature. It should be chosen to allow exploring the whole space rather easily, i.e., with a high acceptance ratio ($\alpha > 0.9$). Then the lower temperatures are chosen such that their histograms overlap to some degree with its closest neighbor, see [48] and [45] for detailed suggestions. One of the criteria for a good temperature ladder is that the swapping step should be accepted with a ratio of approximately 50%. A too small swap rate means an inefficient exchange of information, while a too big swap rate will interupt the local search very frequently that can result in a much blurred image of the target distribution.

While the parallel tempering algorithm is able to accelerate MCMC sampling by increasing the mixing rate, it has a limitation in using multiple processors, namely, we can only use the number of processors that are equal to the number of chains. To prevent the swap rates from being too large we have to limit the number of chains in

total. It has been advised that fewer than ten chains shall be used. That is to say, only ten processors can be fully exploited at most.

3. THE MARKOV CHAIN PRECONDITIONED MONTE CARLO METHOD

The existing parallel MCMC algorithms explored single chain and multiple chain parallel schemes, but the efficiency in terms of parallel computing is very limited. This is due to the sequential nature of MCMC that the proposal of each sample relies on the information of the previous state. It is then attractive to study the *independent Metropolis-Hastings* algorithm which proposes new samples using a distribution $q(x)$ that does not rely on samples for the target distribution $\pi(x)$. The main problem with this idea is that whenever the proposal distribution is not close enough to the target distribution, the proposed samples are most likely to fall in regions of low target probability and the acceptance ratio of such algorithms can be extremely small. This problem becomes more severe when the dimension of the parameter space is high. In such cases, the high probability regions only fill a very small portion of the space.

3.1 The MCPMC algorithm

Inspired by the "interacting Markov chain Monte Carlo" in [14], we derive a new algorithm that is able to remedy this problem. Instead of choosing a proposal function which has no knowledge of the posterior distribution, we construct it by using approximation models of the posterior distribution. For example, we may construct the posterior distribution on a less accurate but much faster solver for the forward problem. When the forward problem is about solving a partial differential equation, we may discretize the equation by using a coarser grid. The posterior built this way will be easier to evaluate, while through a careful treatment we can keep the distribution built on the coarser grid close enough to the accurate model.

Most times, the constructed approximate posterior distribution is again not re-

trievable in an analytic form, therefore we may sample the approximation using some MCMC method. Sampling this distribution is then much faster than sampling the real distribution. While the approximation model is sampling, we randomly select samples from the expanding pool, taking the growing ensemble as a discrete proposal distribution. As more samples are generated from the proxy model, this ensemble should resemble the approximate distribution $\pi_{\text{approx}}(x)$ and the random selection can be considered as if they come straight from an independent draw from $\pi_{\text{approx}}(x)$.

Then, proposed candidates are evaluated independently on different processes to compute an importance weight following equation (2.2)

$$w(x) = \frac{\pi(x)}{\pi_{\text{approx}}(x)}.$$

Comparing the weight of a new proposal and of the current sample, we update the chain for real posterior distribution following the Metropolis-Hastings algorithm. Roughly speaking, this step filters out the samples that have a relatively small value on $\pi(x)$ and replicates the samples whose weight evaluation are large.

Even though the approximate model can be crafted to be close to the accurate model, they are not identical. Thus, samples from the approximate model may not cover the entire high probability regions of the accurate model. Taking this fact into account, we add a step to the new algorithm which uses a short Markov chain with invariant distribution $\pi(x)$ to perturb the samples towards a higher probability area. The last sample obtained from this short chain then replaces the acquired sample. The computation of each short chain is expensive because it may incur several forward model evaluations. But since every such chain can be run entirely in parallel the additional cost can be effectively controlled with a sufficient number of processors.

It is also noteworthy that the preconditioning can be readily extended to multiple levels of approximations rather than having only one approximation $\pi_{\text{approx}}(\cdot)$ and one accurate model $\pi(\cdot)$. For example, to guarantee a smoother transition from the approximate to the accurate, we may insert several intermediate chains whose density functions are weighted geometric means of the approximate and accurate models. That is, we can choose intermediate distributions $\pi_\lambda(x)$ according to

$$\pi_\lambda(x) = \pi_{\text{approx}}^{1-\lambda}(x)\pi^\lambda(x), \qquad \lambda \in (0,1).$$

By carefully choosing a sequence of such temperatures, we will be able to make neighboring distributions close to each other. Refer to [43] for an example of using this setup in a sequential Monte Carlo (SMC) sampling.

As the new algorithm is derived from the interacting MCMC, they share a theoretical basis. Meanwhile, the new algorithm differentiates itself from the original in aspects. The interacting MCMC algorithm is built for state space models where multiple chains are built on parameter spaces with increasing dimensions (i.e., a sequence of *augmenting spaces*). In the state space models, transitions between successive spaces have already been defined as conditional distributions. In our framework, however, there is only one parameter space with a fixed dimension across all the chains. To fit into the same algorithmic framework, we need to build approximation models and sample the approximate distribution as an auxiliary. Since there is no natural transition from the approximate to the accurate, we add the Markov kernel based perturbation step to connect both models. A similar algorithmic modification appeared in [22, 23]. Therein, the authors modified an SMC method which is originally designed for general state space models of increasing parameter dimensions to suit fixed dimension models. Comparing the SMC method in [22, 23], our algorithm

uses Markov chains instead of relying on independently generated samples, hence we can expect better asymptotic stability.

Note that in this algorithm, we use a coarse model to guide the simulation on a fine model, and the Markov chain is mainly used for sampling the approximation distribution, we realize that the coarse model Markov chain can be viewed as a stochastic approximation of the approximation distribution which then serves as a preconditioner for sampling the true distribution. Hence, we name it the *Markov chain preconditioned Monte Carlo* method. In the following, we give the formal description of the algorithm in a general multiple chain version. For conciseness, we show the algorithm in a synchronous way, that is, we assume that for every iteration both the preconditioning chain and the target chain produce exactly one sample. In real implementations, nevertheless, the speed of the two chains depend on factors such as the number of processors we use and the number of perturbations after each update.

Algorithm 3 Markov chain preconditioned Monte Carlo

Suppose there are n chains with $\pi_n(x) = \pi(x)$ and $\pi_i(x), \quad i = 1, \cdots, n-1$ is a sequence of approximations. At iteration k, do:

- For chain $i = 1$, update X_1^k using a traditional MCMC with invariant distribution $\pi_1(x)$,
- For chain $i = 2, \cdots, n$, do the following steps
  - *Select*: randomly propose a sample $X_{i-1}^m, \quad m \in [1, k]$ from the previous chain as X_i^*,
  - *Update*: compute the incremental weight $w_i(X_i^*) := \pi_i(X_i^*)/\pi_{i-1}(X_i^*)$ and make update decision with probability

$$\begin{aligned} \alpha_i(X_i^*, X_i^{(k-1)}) &= \min\left(1, \frac{w_i(X_i^*)}{w_i(X_i^{(k-1)})}\right) \\ &= \min\left(1, \frac{\pi_i(X_i^*)\pi_{i-1}(X_i^{(k-1)})}{\pi_{i-1}(X_i^*)\pi_i(X_i^{(k-1)})}\right), \end{aligned}$$

  - *Perturb*: move X_i^* by a Markov kernel with invariant distribution $\pi_i(\cdot)$ to X_i^k.

This algorithm is well suited for parallel computing in that sample proposals for the i-th chain where $i \geq 2$ can be done in parallel: it is easy to select multiple candidates from the previous chain, evaluate them on different processors (expensive step), make the sequential update decisions and perturb those accepted samples again in parallel. That is to say, all the expensive steps, the before-update evaluation and the after-update perturbation, can be completely parallelized. The speed of the

sampler is also determined by the preconditioning chain on the approximate model. If this chain i) resembles the accurate distribution closely enough and ii) samples the approximate distribution fast, the other chains will acquire good quality candidates swiftly and the convergence will be improved.

3.2 Implementing MCPMC for multiple processors

The MCPMC is designed specifically for Bayesian inverse problems that have the following common features: (a) the likelihood function evaluated on the accurate model is very expensive, (b) the approximate model is much faster to evaluate and (c) yet the evaluation of the approximate model is considerably slower than the necessary communications. Thus, we adopt a tailored master-slave architecture [30] for implementing this algorithm on multiple processors. In a master-slave model, the master generates tasks and allocates them to the slave processes. In particular, suppose there are a total number of N processes, we assign process 1 for the preconditioned Metropolis-Hastings chain using the approximate model, process $2, 3, \cdots, N-1$ to be real slaves that repeatedly evaluate samples, using the accurate model, from and return the evaluations back to the master process, and process 0, the master process, to coordinate all the remaining tasks which include random selection, update decision, perturbation decision, data structure management and data communication.

The communication between processes 0 and 1 is one way—only the base chain has to send new sample candidates to the master process. On the other hand, the communication between the master and its real slaves are both ways: the master prepares tasks—it could either send a sample for a single evaluation on the target model so as to compute the weight of that sample, or it could send a sample for the perturbation step which is in fact a short path of MCMC on the target model—and

continuously checks whether slaves have sent back results for evaluations. Whenever a slave finishes a task, it will send some feedback—a number—back to the master and the master will hand a new task over to it. See Figure 3.1 for an illustration of our implementation for the algorithm model.

The reason for not merging the tasks of process 0 and 1 into one process is that it is impossible to deal with any communication needs when a process is computing an approximate model evaluation. In contrast, our design ensures that the master process is always responsive: all tasks done by this process only take a fraction of the time of even the approximate model evaluation, therefore whenever a slave process finishes its current job and demands a new task, the master will be able to make the designation immediately. Simply put, separating the base chain simulation from the central commanding avoids the potential communication overhead in the whole algorithm.

Note also that this design can be generalized in a relatively straightforward way to multiple chain situations—instead of having one master, we can simply have as many masters as there are MCPMC chains (not including the base chain). The only additional layer of communication would be one master sending new samples continuously to the master of the next chain, which is essentially the same as the type of communication between processes 0 and 1 mentioned above.

3.2.1 Discussion on the parallel efficiency

We first analyze the scalability of the algorithm in terms of producing fine level samples. Suppose the time spent for evaluating a sample on the fine model is t_{f}, and the time for communication, i.e., sending a sample between processors is t_{comm}. In most of the cases we are interested in where t_{f} is approximately the time for solving a partial differential equation on a mesh that is fine enough to resolve certain details,

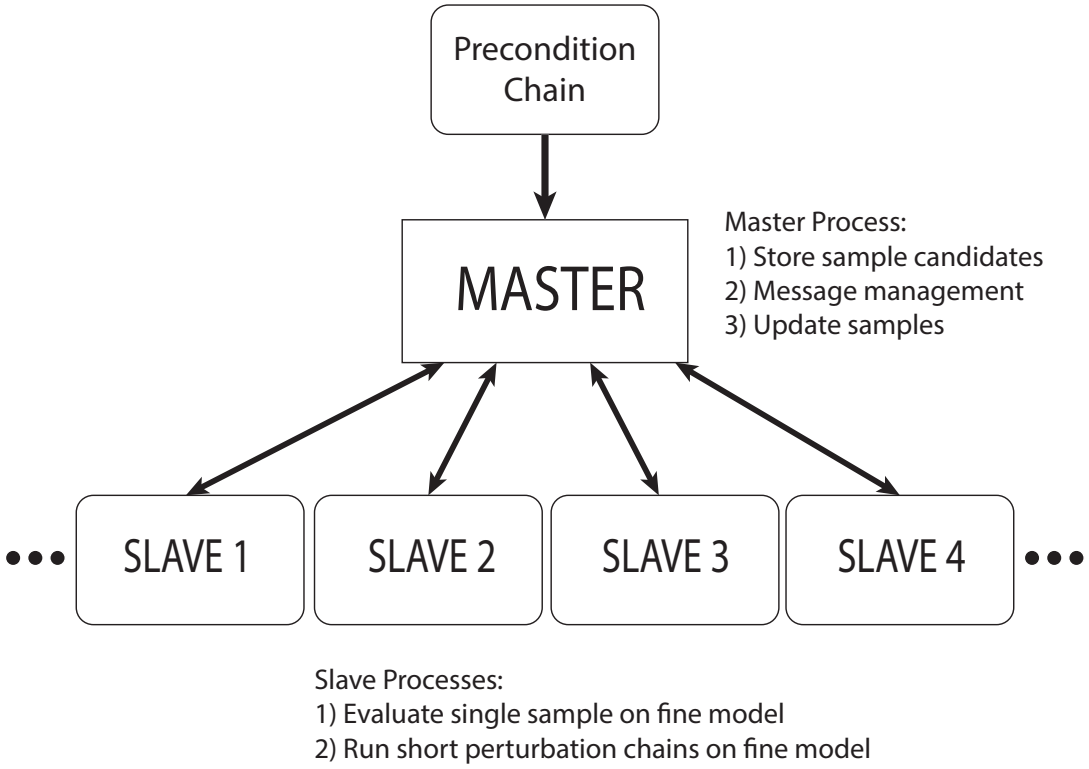


Figure 3.1: A graphical illustration of the modified master-slave architecture with which we implement the MCPMC algorithm.

we can always assume that $t_{\mathrm{f}} \gg t_{\mathrm{comm}}$. If we define the speedup (sp) as the rate of samples produced by MCPMC divided by the rate using a fine Metropolis-Hastings algorithm, N as the number of processors available and p as the number of steps from the short perturbation Markov chain happening at each step, we would then have $N-2$ slaves repeatedly working on the single evaluation for update decision and the p step perturbation. Therefore, we obtain that

$$\mathbf{sp} = \frac{t_{\mathrm{serial}}}{t_{\mathrm{parallel}}} = \frac{(N-2)t_{\mathrm{f}}}{(1+p)t_{\mathrm{f}}} = \frac{N-2}{1+p},$$

and the efficiency (eff) of the algorithm is

$$\mathrm{eff} = \frac{\mathbf{sp}}{N} = \frac{N-2}{N}\frac{1}{1+p} \to \frac{1}{1+p}$$

when $N \to \infty$. Recall that the efficiency of the prefetching algorithm is $\ln N/N \to 0$, therefore the MCPMC is guaranteed to outperform the prefetching in obtaining more samples when many processors are available.

It is however necessary to realize that getting more samples is not always equal to converging faster to the posterior distribution. An extreme counterexample will be that if we have N very large that we can simultaneously process many fine level evaluations when only one coarse level sample is produced, we then are faced with the situation that all the fine level evaluations are performed only on an identical sample at the beginning of the sampling process, which is no better than obtaining only one fine level sample. On the other hand, if the sampler has run long enough and the preconditioning base chain is close to $\pi_{\mathrm{approx}}(x)$, the approximating distribution, the randomly selected samples are then nearly independent samples from $\pi_{\mathrm{approx}}(x)$. Henceforth, the speedup can go over the sheer speedup in number of samples because

such an independent sampler with certain regularization property between the target and proposal distributions may bring a faster statistical convergence over the random walk based Metropolis-Hastings algorithms [50].

To have an estimate of the speedup taking statistical convergence into consideration, we redefine the speedup using the time needed to obtain a certain number K of equivalent independent samples where statistical independence is defined with respect to some function $h(x)$:

$$\widehat{\mathbf{sp}} = \frac{\hat{t}_{\text{mh}}}{\hat{t}_{\text{mcpmc}}} = \frac{2K\tau_{\text{mh}}(h)t_f}{2K\tau_{\text{mcpmc}}(h)(\frac{p+1}{N-2})t_f} = \frac{\tau_{\text{mh}}(h)(N-2)}{\tau_{mcpmc}(h)(p+1)}, \tag{3.1}$$

where, for $* = \text{mh}$ *or* mcpmc, $\hat{t}_*$'s are the scaled time for getting the number of samples that are equivalent to K independently drawn samples, τ_* are the integrated autocorrelation times that are defined in (2.5), and we have adopted the effective sample size formula (2.6). Equation (3.1) explains the situations mentioned earlier well: at the beginning of MCPMC sampling, due to limited size of the preconditioning chain, most selected samples are identical and hence τ_{mcpmc} can be very large, which leads to a small $\widehat{\mathbf{sp}}$, whereas the integrated autocorrelation time can be dramatically reduced if the base chain is already long enough and the proposal is as if they were generated independently from the approximating distribution—this is where the $\widehat{\mathbf{sp}}$ can grow even larger than $\mathbf{sp}$. As a remark, it is worth noting that equation (3.1) also hints at the reason that we want the approximating distribution to be close to the target posterior distribution: if it were not the case, or, equivalently, if the variance of $w(x) = \pi(x)/\pi_{\text{approx}}(x)$ was large, MCPMC would frequently hit some sample that has a large $w(x)$ and reject many samples after it, resulting in a large τ_{mcpmc} and a small $\widehat{\mathbf{sp}}$. As we can show in the two inverse problem cases in the following chapters, even for relatively high dimensional, complex models, we are able to build

the approximating distribution close to the target so that a considerable portion of the proposed samples get accepted.

Example 3.2.1. *As an example, let us compute the speedup* $\widehat{\mathbf{sp}}$ *using the data from the optical tomography example in chapter 5. Without loss of generality, we choose the function* $h(x)$ *to be the* L_2*-norm of each sample, and we compute the truncated version of IACT*

$$\tau_k(h) = 1 + 2\sum_{i=1}^{k} \gamma_j(h)$$

in which we chose $k = 3000$*, a lag at which we observe that the autocorrelation for the Metropolis-Hastings is small. Given the fact that the autocorrelation function drops much faster for the MCPMC algorithm, our estimate is relatively conservative.*

The resulting IACT's are

$$\tau_{mcpmc} = 10.91 \quad \textit{and} \quad \tau_{mh} = 72.83.$$

Also, choosing the number of processors $N = 50$ *and the number of perturbations* $p = 5$*, we can calculate that* $\widehat{\mathbf{sp}} = 53.35$*. Namely, under the measure of obtaining equivalent independent samples, the MCPMC achieves super linear speedup. As a remark, we may achieve a larger speedup by using more processors.*

3.3 Example: A two dimensional Gaussian distribution

We first study several important details that can guarantee a successful implementation of MCPMC and make the MCPMC sampler more efficient. In the following example, we find that it is imperative to have neighboring distributions "close" enough to each other, that is, they should have significant overlaps. We also test the difference of convergence using variable number of perturbations and conclude that it is beneficial to increase the number of perturbation steps per iteration but only to

a certain threshold.

We study the use of MCPMC through a two dimensional Gaussian distribution example. The target distribution has a mean vector $(5, 0)$ and an identity covariance matrix. We take another Gaussian distribution with mean $(0, 0)$ and an identity covariance as the approximate distribution. The first chain proposes samples depending on the previous state and updates using a Metropolis Hastings (MH) method. The subsequent chains use MH as the Markov kernel to perturb samples acquired from the previous chain.

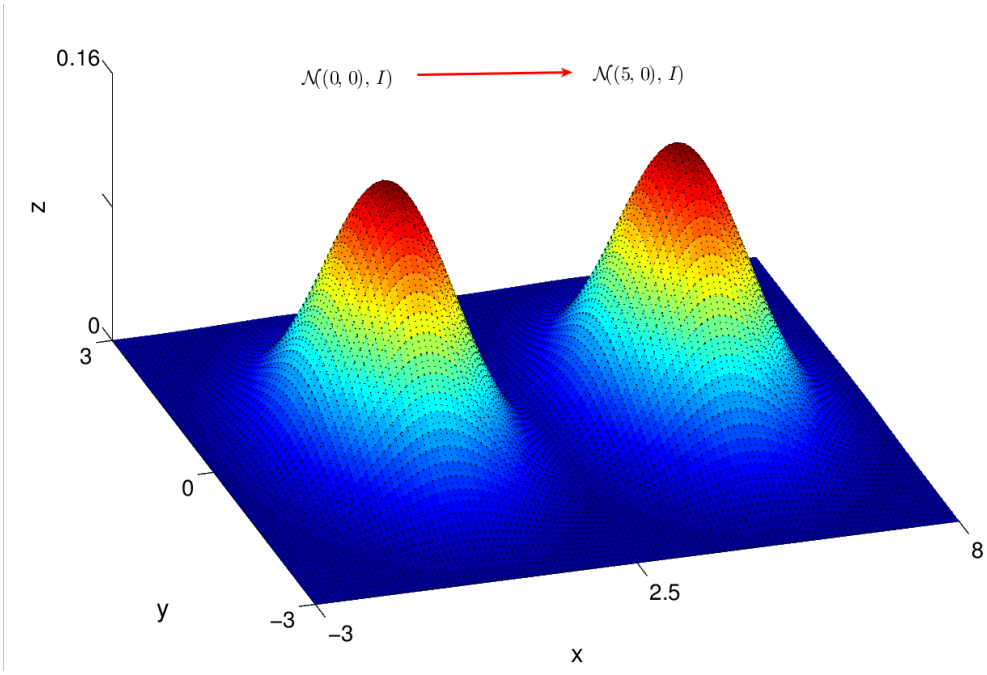


Figure 3.2: An illustration of the two dimensional Gaussian distribution problem where we try to sample the distribution centered at $(5, 0)$ using a MCPMC with the preconditioning chain being an MCMC sampling the "approximate" distribution centered at $(0, 0)$.

Distribution overlap: It is important that neighboring distributions, i.e., the

distributions for adjacent chains be close enough to each other to guarantee efficient updates. When their modes are distant compared to their variances, the overlap between them is small. This way, the variance of the incremental weights $w(x) = \pi(x)/\pi_{\text{approx}}(x)$ is significant which then could result in a poor estimate because the variance of the sample estimate is proportional to $1 + \text{Var}\,(w(x))$ [48].

As is shown in Figure 3.2, the approximate Gaussian distribution is separated from the target distribution. We were unable to directly sample the target from the approximation after one million samples. The poor statistical property of the two-chain setting is indicated by the low acceptance ratio at the update steps for the second chain as is shown in Table 3.1. To improve the situation, we insert $N-2$ intermediate chains to narrow the differences between neighboring distributions. These chains all have normal distributions of the identity covariance matrix and a center at $(5i/(N-1), 0)$, for $i = 1, \cdots, N-2$. These additional chains build a smooth transition, making it easier to move samples from one distribution to the other. The increased average acceptance rate, average taken over all N chains in the update steps with $N = 2, 3, 5, 7, 9$ and 11 supports this conclusion.

Number of chains	Center distance	Average update AR
2	5.0	0.002
3	2.5	0.097
5	1.25	0.38
7	0.833	0.56
9	0.625	0.66
11	0.5	0.72

Table 3.1: Average acceptance ratios (AR) in the update stage when using different numbers of chains. The "center distance" column shows the distance between mean vectors of neighboring distributions.

Number of perturbations: Another factor that determines how fast the sampler converges is the number of perturbation steps. The perturbation is a short Markov chain based only on the accurate distribution $\pi(x)$, which means that such a step could move the sample towards a higher probability region under $\pi(x)$. Hence, taking more steps in the short chain might move the sample further and lead to a better sampling quality with the same number of samples on the last chain.

Figure 3.3 shows a comparison of convergence using different numbers of perturbation steps at each iteration. When we only use one perturbation at each iteration, the convergence behavior of the sampler is poorer: the error barely decreases after twenty million samples. Every unit increment of the number up till four per iteration has given substantial improvement in convergence. The difference between using four and ten perturbations becomes much smaller. Considering that these perturbation steps are sequential and each step evaluates the forward problem, this suggests that we should not increase the number too much so as not to increase the computational burden.

3.4 Example: A multi-modal Gaussian mixture distribution

In this multi-modal example, we demonstrate the potential power of the new algorithm in increasing the sampler's statistical efficiency. When a probability distribution has more than one mode, it is possible that a sampler gets trapped at one or a few of these modes. That is, multi-modal distributions can impede the ergodicity of a sampler, making it converge extremely slowly. An effective solution to this local trap problem is to use a sequence of tempered distributions that are defined at (2.7). For this tempered scheme to work, the sampler needs to have an infrastructure that runs multiple chains at different temperatures, and there should be a way for these chains to interact. MCPMC serves these purposes. We can assign the highest tem-

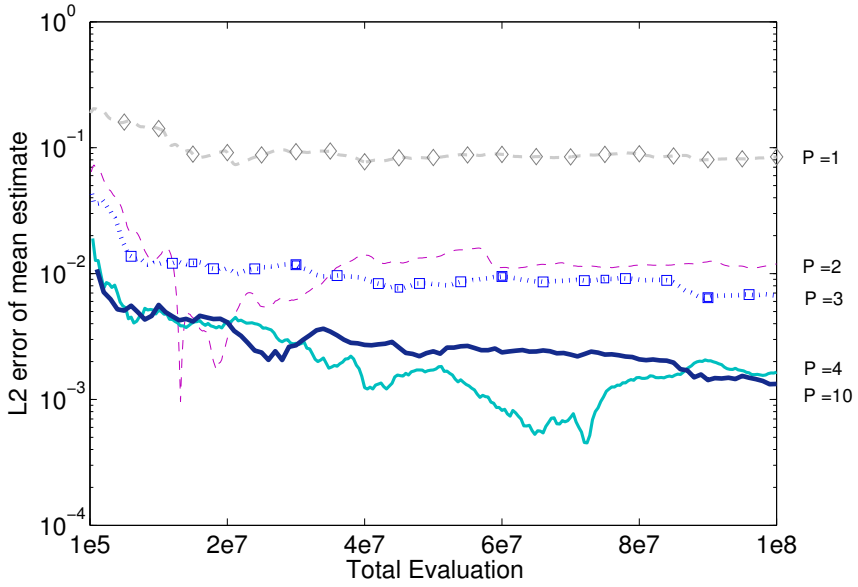


Figure 3.3: Convergence comparison of samplers with different perturbation steps. The x-axis is the total number of perturbation evaluations taken on the accurate model. So, for the sampler with $P = 10$, the total number of samples shown in this plot is 10^7 because obtaining each sample requires running a short chain of 10 steps. On the contrary, the sampler with $P = 1$ has 10^8 samples shown in the plot. Even so, samplers with more perturbations demonstrate a better convergence.

perature to the base chain, such that it can easily sample around the entire sample space. The subsequent chains will use a ladder of decreasing temperatures, which helps gradually filter out the over scattered samples.

We design this experiment following the similar experiment in [46]. To illustrate, we use a mixture of two dimensional Gaussian distributions

$$\pi(x) = \sum_{i=1}^{N} \omega_i \mathcal{N}(x_i, \sigma^2),$$

where $N = 10$, ω_i's, the weights, are all set to be 0.1, $x_i \in \mathbb{R}^2$ are ten centers randomly drawn from the square $[0, 10] \times [0, 10]$, and σ^2 are set to be 0.1 for all the Gaussian modes. We sample this distribution both with a random walk Metropolis-Hastings (MH) chain and an MCPMC chain, each with 1000 samples. For the MH chain, we set the random walk step size to be 1.0 for both components, which gives an acceptance ratio of 0.27, showing that it is reaching optimal performance. For the MCPMC chain, we utilize a total of 6 chains of decreasing temperature that follows the formula

$$T_i = a^{b(6-i)}, \tag{3.2}$$

where T_i is the temperature for the i-th chain and $a = b = 1.5$ are constants that we tune such that the neighboring distributions are close to each other. The samples from the sixth chain ($T_6 = 1$) are taken as the final sampling results.

In Figure 3.4 we show the sampling results from both the MH and the MCPMC samplers. The MH sampler is trapped at the lower left corner where two Gaussian modes reside. On the contrary, the MCPMC successfully samples all ten modes. Though there seems to be fewer samples at the left lower corner modes, it is mainly due to the small number of samples and we do observe a better balance with more

points sampled. Also, in this figure, we plot the sample paths for the sixth chain of the MCPMC algorithm, from which it is clear that the sampler achieves the ergodicity not by chance. During its process, the sampler frequently jumps between different modes of the distribution, thus showing that it has very good mixing properties. This is somehow straightforward to understand because the random selection step in MCPMC naturally breaks the confinement of local search, which is typical in pure random walk type samplers.

We also show the samples from all chains of the MCPMC algorithm in Figure 3.5. The first chain has largest variance and samples the entire domain relatively easily. Therefore, it provides a robust 'preconditioner'—a pool of sample candidates that covers all regions of interest to feed the subsequent chains. With the help of importance weight updates from each chain to its right neighbor, the sampler is able to gradually filter out the samples that posses small probabilities in the true distribution. The additional perturbations then further move samples towards important regions of the true distribution.

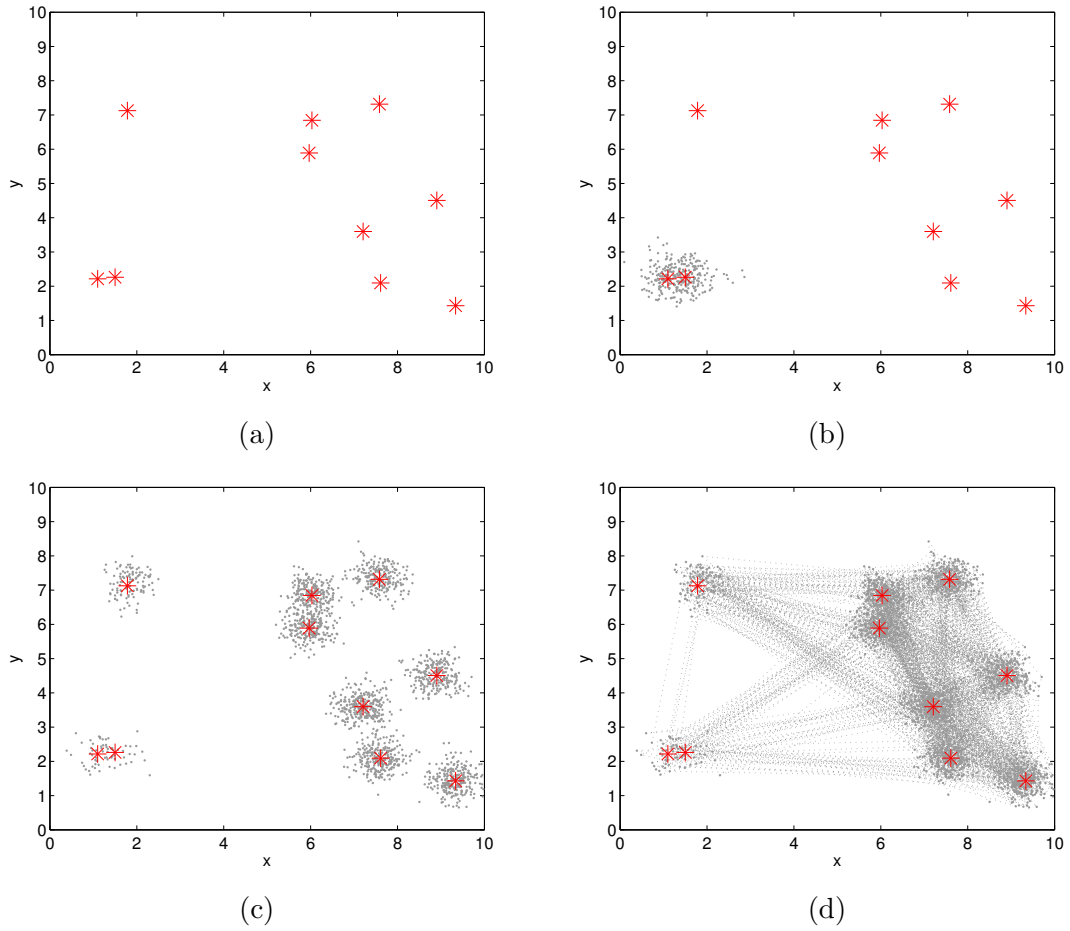
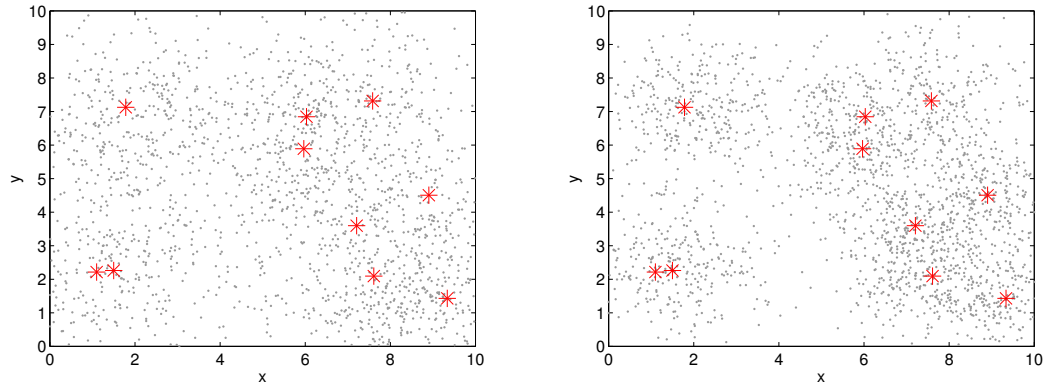
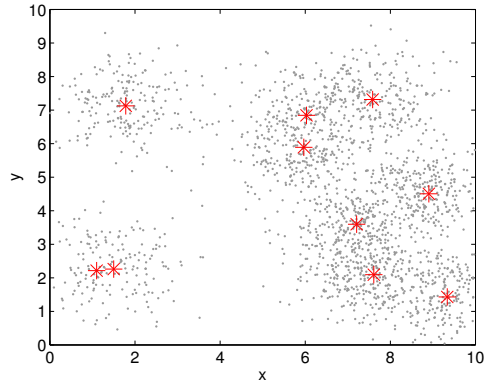


Figure 3.4: Sampling the two dimensional Gaussian mixture distribution: (a) shows the centers of the ten modes; (b) shows the result of 1000 samples from an MH sampler; (c) shows the result from the MCPMC with 6 chains and 1000 samples on the sixth chain; (d) shows the sample paths for the same samples as in (c).

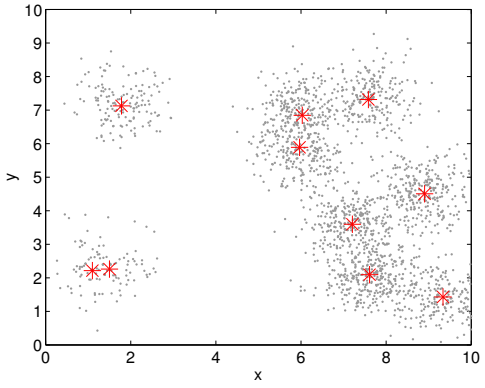


(a) Chain 1

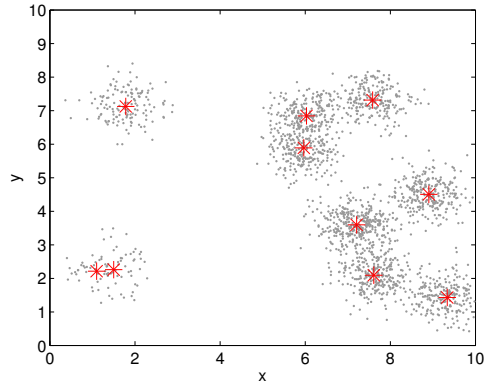
(b) Chain 2



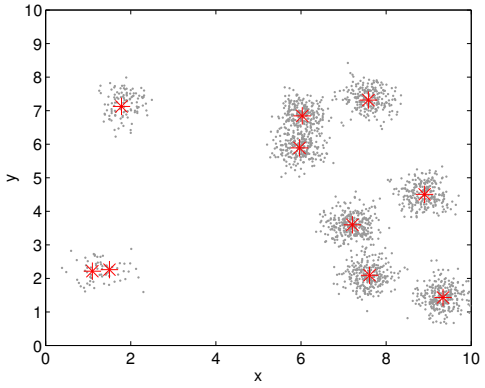
(c) Chain 3



(d) Chain 4



(e) Chain 5



(f) Chain 6

Figure 3.5: 1000 samples from the first chain to the sixth chain of the MCPMC algorithm. The temperature decreases with respect to $i, i = 1, 2, \cdots, 6$. The i-th chain serves as a preconditioner for the $i+1$-th chain which fetches samples from the preconditioner and filter samples according to its own probability distribution.

4. APPLICATION: AN INVERSE PROBLEM WITH ELLIPTIC EQUATIONS

4.1 Problem description

In this chapter, we investigate how to sample the posterior distribution for an inverse problem with MCPMC. The problem considered herein has an elliptic partial differential equation as the forward model:

$$-\nabla \cdot (x(z)\nabla p(z)) = q(z), \quad \forall z \in \Omega,$$

$$p(z)|_{\partial\Omega} = g(z).$$

The inverse problem is to infer the parameter field x from observations of the pressure at several locations in Ω. This forward model has direct applications in nuclear waste disposal problems [60] and is the basis for problems arising in electrical impedance tomography [18] and multiphase flow problems [25]. We implement the solver of the forward problem in the finite element library deal.II [6, 7].

In our experiment, we use a reference coefficient field to generate synthetic data—data that are taken as reference physical measurements. We use the reference field x_{ref} as shown in Figure 4.1, input it in the forward model, and use an adaptive finite element method [9] to accurately solve for p which is shown in Figure 4.1. Then, we evaluate this solution at 361 locations equally distributed in the unit square domain. These consist of the physical measurements in this example.

We adopt a pixel type discretization for the parameter $x(z)$: we discretize the parameter field with an 8×8 grid and use piecewise constant functions for the approximation. For the likelihood function, we assume a Gaussian additive noise

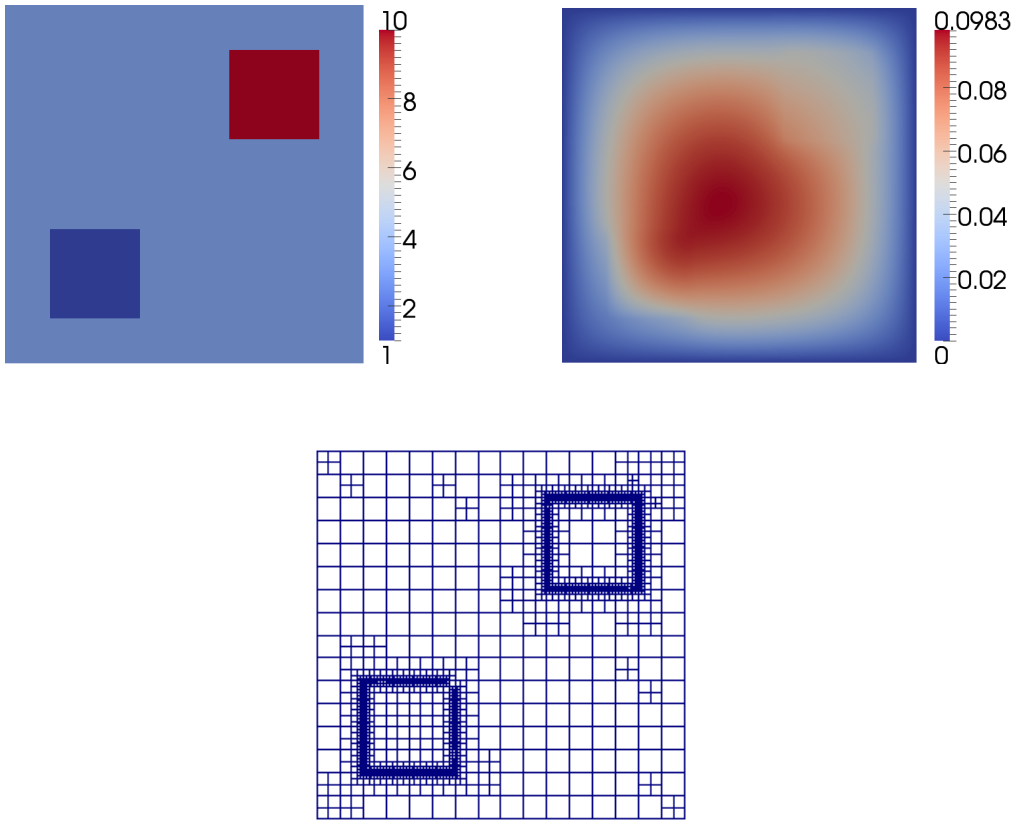


Figure 4.1: Generation of the synthetic data. The top left figure shows the reference coefficient field $x(z)$, the top right figure shows the solution $p(z)$ using the reference coefficient and the bottom figure shows the adaptively refined mesh for obtaining the solution.

model

$$y = f(x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \Sigma_d),$$

where $y = \{y_1, y_2, \cdots, y_{361}\}$, $f : \mathbb{R}^{64} \to \mathbb{R}^{361}$ is the parameter-to-observation map that predicts the discrete measurements from solving the elliptic equation with coefficient x, and Σ_d is the data covariance matrix. We also assume, in all experiments, that the noise level is 1%, i.e. the standard deviation σ is chosen to be one percent of the maximal magnitude of all the synthetic data obtained from the reference coefficient:

$$\sigma := 0.01 \max_{i=1}^{361} |f(x_{\text{ref}})_i|.$$

Choosing proper priors for Bayesian inverse problems is an active research area. One of the frequent options is the Gaussian prior distribution. If the measurement is also a Gaussian distribution and the parameter-to-observation map is linear, the posterior distribution is also Gaussian and it is straightforward to compute the mean and covariance matrix. Another example would be l_1 type priors such as the total variation norm prior or the Besov priors that can promote sparsity and hence are widely applied in imaging. We show some random samples from the total variation distribution in Figure 4.2 in which pixels tend to correlate with their neighborhoods. In Figure 4.3, we show random samples from a Gaussian prior distribution where the pixels are independent of each other, showing random perturbations from the mean vector. More advanced Bayesian modeling techniques such as hierarchical modeling has also been studied for Bayesian inverse problems to achieve autonomous hyperparameter selection. In this example, we opt for a simple choice of the prior so that we can focus on comparing the efficiency of different samplers rather than also blending it with modeling effects. For the pixel type discretization, we choose the

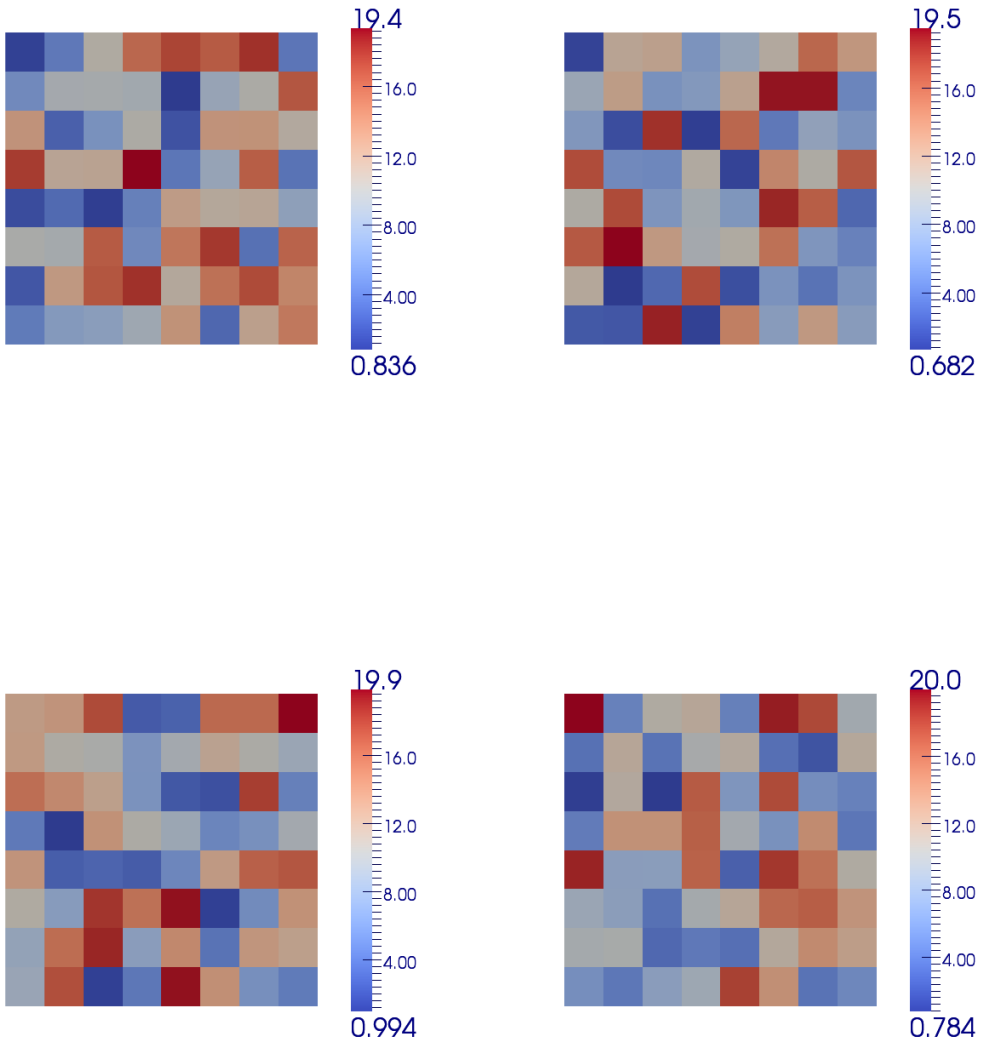


Figure 4.2: Four samples from a total variation prior distribution. Pixel values vary across the range of parameters and the pictures are blocky.

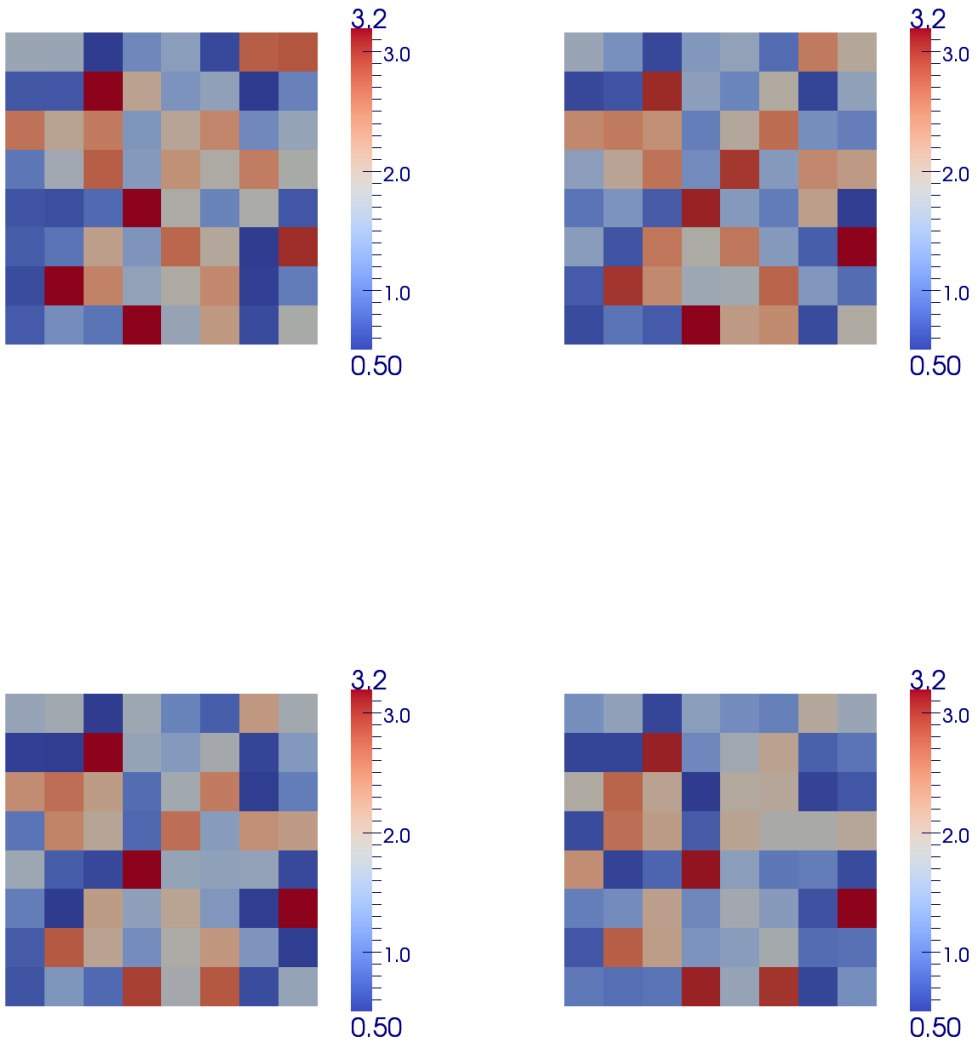


Figure 4.3: Four samples from a Gaussian prior distribution. Pixel values are independent of the values of their neighbor.

Gaussian prior

$$\pi_{\text{prior}}(x) \propto \exp\left(-\frac{||x - x_0||^2_{\Sigma_m^{-1}}}{2}\right),$$

where x_0 is some prior guessed value of the parameter and Σ_m is the model covariance matrix. Here, we choose $x_0 = 0$.

Having set up the likelihood and the prior, and by further assuming that

$$\Sigma_d = \sigma_d I_d \quad \textit{and} \quad \Sigma_m = \sigma_m I_m,$$

we now have the posterior distribution as

$$\pi(x|y) \propto \exp\left(-\frac{1}{2}\frac{||y - f(x)||^2}{\sigma_d^2} - \frac{||x||^2}{2\sigma_m^2}\right).$$

We solve the forward problem using linear finite elements [12, 26] on two meshes Γ_h and Γ_H. The fine grid Γ_h for sampling algorithms is chosen to be the uniform mesh of size 64×64 and the coarse grid Γ_H is of size 16×16, as are shown in Figure 4.4. The posterior distribution obtained through a forward model on the fine mesh is considered as the accurate distribution and the posterior distribution built on the coarse model is the approximating distribution. We sample the fine distribution with a plain Metropolis-Hastings algorithm and compare the results to an MCPMC sampling using the coarse grid samples as a preconditioner. Note that in this computation the difference of two posterior distribution evaluations only lies in the likelihood function and the choice of the prior in fact does not interfere with the comparison. Therefore, we choose σ_m so that the prior distribution is flattened compared to the likelihood function, hence reducing the impact of the prior distribution.

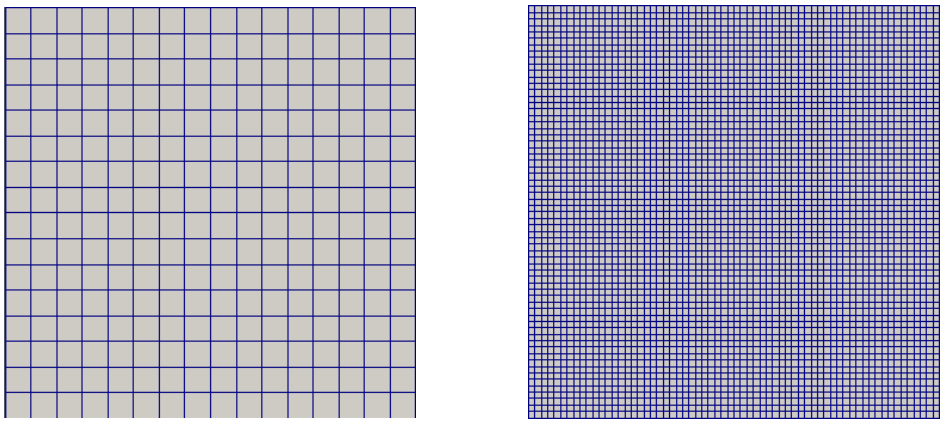


Figure 4.4: The meshes used for the approximate model (left) and the accurate model (right).

4.2 Enhanced error model

Some preliminary runs of the experiment indicate that running MCPMC directly with the coarse grid samples as a preconditioner is inefficient—the update acceptance ratio of the importance sampling step on the fine chain is below 0.001. As has been shown in the Gaussian distribution case, a low acceptance ratio associates with poor overlap between the approximating and the accurate distributions. Admittedly, even if the approximate model and the accurate model are distant we are able to mend the system by inserting extra chains with tempered distributions that bridge the ends, this is however never an optimal strategy because additional chains will demand extra computing resources. Hence, we seek feasible techniques to narrow the gap between two distributions so that the two chain scheme can also achieve efficiency. In the following, we discuss the *enhanced error model* which achieves this purpose with little additional computation.

The conventional Gaussian additive noise model [39] constructs a likelihood function which assumes that differences between ideal predictions $f(x)$ and actual mea-

surements y result from an error model of the form

$$y = f_i(x) + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \Sigma_d)$$

for both the fine grid model f_2 and the coarse grid model f_1. The assumption underlying it is that the error term ϵ includes all noise sources such as measurement errors, model uncertainties or the numerical discretization inconsistency. Though, surprisingly, this simplification has worked well when solving the inverse problems on a fixed mesh by using either an optimization method or a stochastic sampling, it is over simplified as an approximate model which solves the forward problem on a coarse mesh.

On one hand, the discrepancy between a fine mesh solution and a coarse mesh solution is not likely to have a zero mean. Nor do the estimate errors between these two models behave by any means uncorrelated. The conventional model is then surely inappropriate for this situation. On the other hand, such a discrepancy can be quantified as long as one can simulate both the coarse and fine grid models.

In previous work [1, 39, 42], this was overcome by splitting the single term noise into two parts: one represents the discretization error coming from different mesh resolutions, and the other term still incorporates all other categories of uncertainties. The new model is defined as

$$y = f_2(x) + \epsilon = f_1(x) + (f_2(x) - f_1(x)) + \epsilon.$$

In this refined model, the new ϵ contains all the noise except for the discretization error between the grids. We can model $\epsilon \sim \mathcal{N}(0, \Sigma)$ as before, but for the new uncertainty term $\eta = f_2(x) - f_1(x)$ we assign it another normal distribution $\eta \sim$

$\mathcal{N}(\mu, \Sigma_1)$. To estimate the mean and covariance for η, we pre-simulate an ensemble of N samples $\{x_i\}_{i=1}^N$, put each x_i in both the coarse and fine forward solvers to compute the differences in measurements and summarize an estimation with:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \left(f_2(x_i) - f_1(x_i) \right),$$

$$\Sigma_1 = \frac{1}{N} \sum_{i=1}^{N} \left(f_2(x_i) - f_1(x_i) - \mu \right) \left(f_2(x_i) - f_1(x_i) - \mu \right)^T.$$

The enhanced error model is a nice example of the advantages of statistical inverse problems over their deterministic relatives—the uncertainty can be separated to adapt different types of errors. For those errors that only result from computer simulations, we are able to quantify them through an offline computation. This way, the model can have reasonably better accuracy. We demonstrate such an improvement through simulations for the elliptic problem. We run MCMC chains with 1,000,000 samples on a 16×16 grid using both the coarse approximate model and the enhanced error model and compare the results with another simulation of the same number of samples on a 64×64 grid. Figure 4.5 shows the error between the mean estimation and the reference (as discussed in Section 4.1) for all 64 pixels. At most pixels, the enhanced error model is able to obtain a better estimate than the coarse grid model, indicating that the mean value correction takes positive effect on the sampling quality. Figure 4.6 shows the histogram estimate of the marginal distributions at three pixels (x_5, x_{34} and x_{51}). While the histograms from the coarse grid model deviate from those of the fine grid, the EEM estimation overlap well with the fine for all three instances.

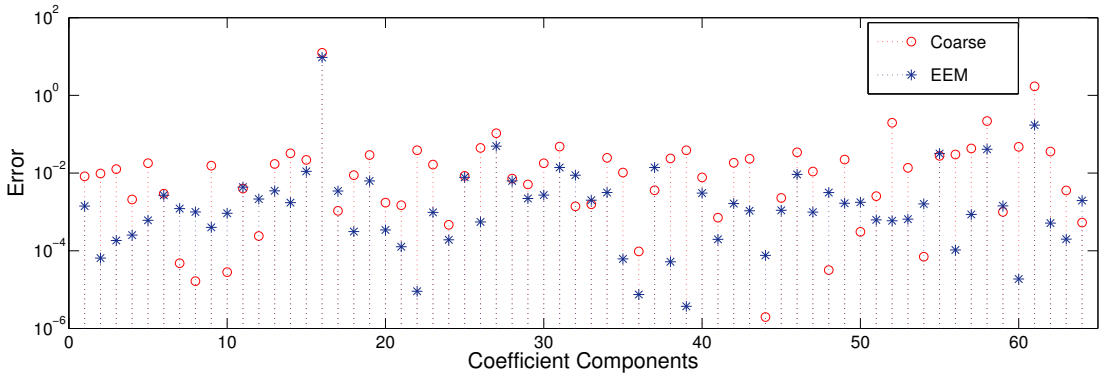


Figure 4.5: Error comparison for a plain MCMC algorithm with either a coarse grid model or an EEM using one million samples. The x-axis is the 64 pixels, i.e., the components x_i of the parameter which appears as a coefficient in the elliptic equation. The y-axis is the error $|\bar{x}_i - x_i^*|^2$, i.e., the square error between the mean estimator and the real value.

4.3 Numerical results

Now let us return to discussing the efficiency of sampling the model inverse problem. As a reference, we first run a Metropolis-Hastings (MH) chain on the fine level grid for 2,000,000,000 samples—a large enough sample number with which we believe that the chain has converged. Out of this reference ensemble, we computed the mean vector x_* and the covariance matrix C_* and compute, for any ensemble of samples $\{x_i\}_{i=1}^N$ in later experiments, the weighted error according to the definition in (2.4):

$$\text{error}_N = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_*)^T C_*^{-1}(x_i - x_*).$$

The reason for comparing with the average out of a reference sampler instead of comparing directly with the synthetic parameter field is that after adopting the prior distribution we can not expect the reconstruction to be exactly the same as the synthetic parameters; rather, only a long enough MH sampler can provide the

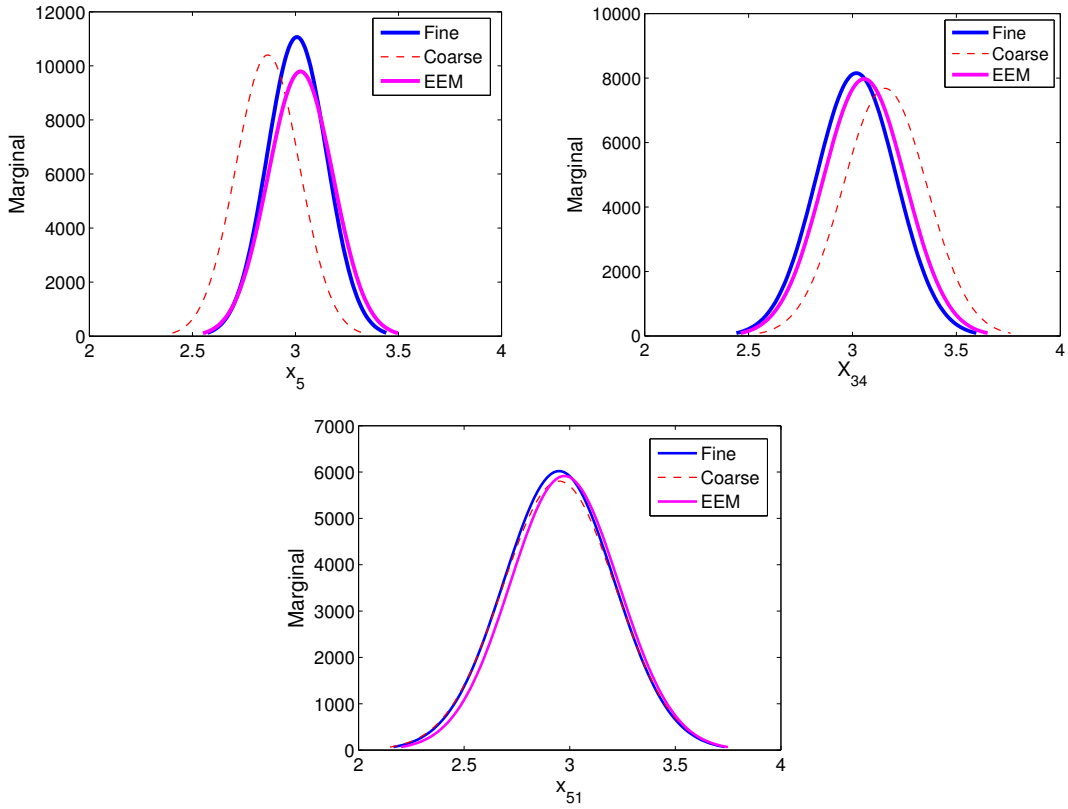


Figure 4.6: Using the same data as above, the figure shows a comparison of the histogram approximated density functions for the marginal distribution of three coordinates: x_5, x_{34} and x_{51} between the coarse grid, the fine grid and the enhanced error models.

real mean vector for the posterior distribution. In addition, by weighting the error with the inverse of the ensemble covariance matrix C_*, we can hope to eliminate the impact of some components that are not inferable with the sampler. For example, in the current experiment, the pixels near the center can be estimated very poorly because the gradient of the velocity is close to zero and a large range of parameter values at these pixels can lead to the same measurements. While such phenomena are inherent due to the ill-posedness of inverse problems, we intentionally reduce their impact so that our comparison focuses more on the efficiency of different sampling algorithms.

We make comparison between the results with a coarse chain MH, a fine chain MH and an MCPMC sampling. The coarse discretization has a total of 256 cells and 289 degrees of freedom (DOFs), while the fine discretization has 4096 cells and 4225 DOFs. To reduce the gap between probability distributions for both chains, we employ the enhanced error model discussed above by precomputing 3000 samples on both the coarse and the fine meshes and compute the mean and covariance of the Gaussian approximation error model with measurements obtained from these samples. Also, the MCPMC uses ten perturbation steps to increase statistical efficiency.

We show the numerical results obtained from running the MH algorithm on a single processor and the MCPMC with 12 processors. In Figure 4.7, we plot the conditional mean estimator for samples from both samplers. Both successfully estimate locations and values of the targets correctly. As is already mentioned earlier, the observable artifact on the one central pixel may be due to the fact that it is impossible to infer the value of the coefficient where ∇u is nearly zero, which can be seen from the solution in Figure 4.1. We plot convergence curves for all three samplers in Figure 4.8. From the plot it is clear that the MCPMC has the same accuracy as the fine mesh MH algorithm, whereas the coarse level MH sampler gives

a larger asymptotic error, because the coarse level parameter-to-measurement map $f_1(x)$ does not adequately represent the true map. This fact proves the capability of the MCPMC algorithm to move from the proposal distribution to correctly sample the posterior distribution. We also estimate the efficiency of both samplers in terms of computing time. After repeatedly running coarse and fine evaluations for 20 times, each running 100 samples, we obtain that the average evaluation time for a single sample on the coarse mesh is 0.011 second and is 0.084 second on the fine mesh. Also, through comparing the sample numbers of both the coarse chain and the MCPMC chain in the same experiment, we estimate that the fine target chain is at a speed of about 80% of the coarse base chain in the given combination of processor number and perturbation steps. In Figure 4.9, we plot the error scaled by the estimated time of computing. The MCPMC manages to converge faster than the fine level MH chain at any time, while after about 2000 seconds it shows a better convergence than the coarse level base chain. It is conceivable that increasing the number of processors will further increase the convergence speed of MCPMC.

Note that there is no direct comparison between the efficiency of a multiple chain scheme such as the parallel tempering and the MCPMC because they accelerate the sampling process in different ways. While parallel tempering increases mixing rate, the MCPMC allows multiple sample evaluation simultaneously. It is also noteworthy that the MCPMC can always adopt parallel tempering when sampling the approximation distribution, or at the perturbation steps. Such a hierarchical scheme should further increase the sampling speed.

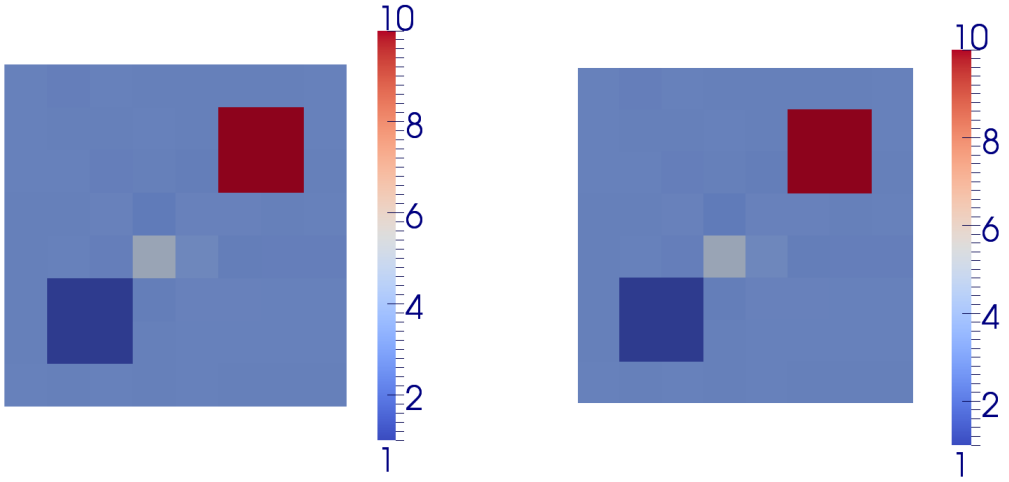


Figure 4.7: Left: Conditional mean reconstruction using 10^6 samples from the MH sampler on the fine grid. Right: Conditional mean reconstruction using 10^6 samples from the MCPMC chain that evaluates on the accurate model.

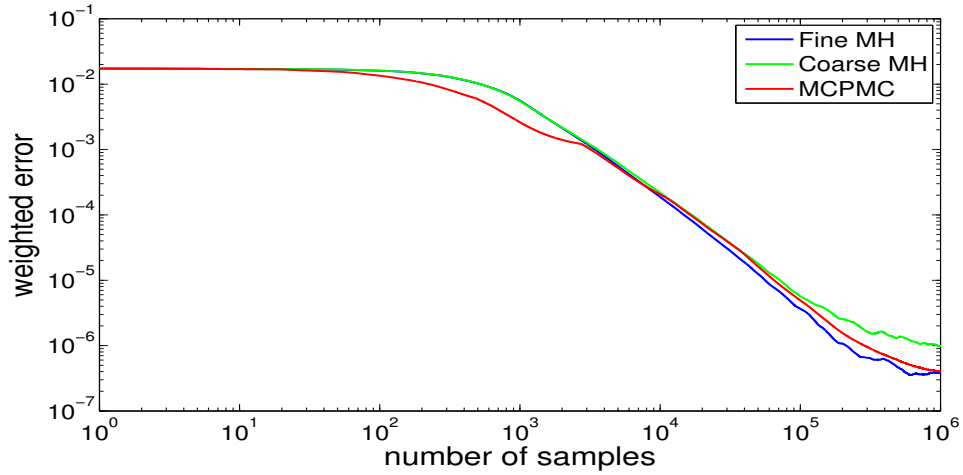


Figure 4.8: Sampling error as defined in (4.3) for the coarse level MCMC (green), fine level MCMC (blue) and MCPMC (red).

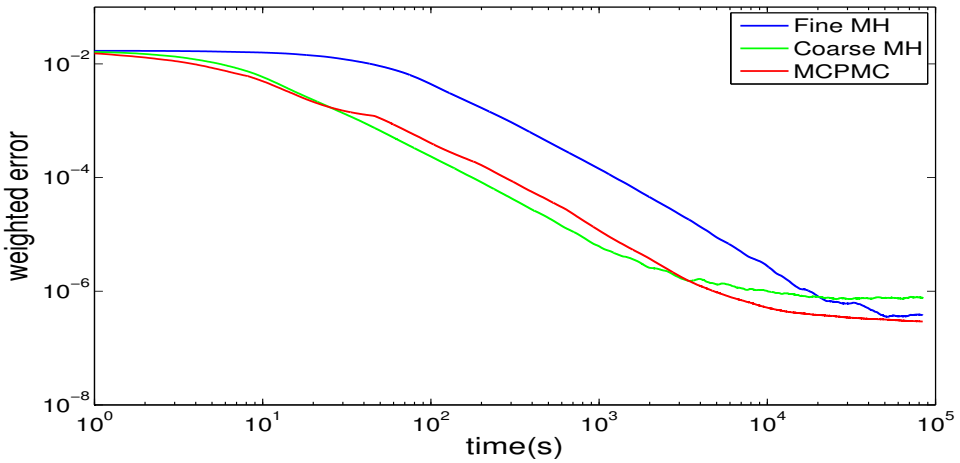


Figure 4.9: Sampling error plotted as a function of estimated running time for the coarse level MCMC (green), fine level MCMC (blue) and MCPMC (red).

5. APPLICATION: THREE DIMENSIONAL OPTICAL TOMOGRAPHY

To further illustrate the strength of the Markov chain preconditioned Monte Carlo algorithm, we apply it to uncertainty quantification to a fluorescence enhanced optical diffusion tomography (FDOT) problem. Optical tomography is an emerging medical imaging method which recovers the spatially variable tissue properties inside a subject using measurements of light intensities on its boundary. It has important applications to breast and cervix cancer detection and staging, lymph node imaging as well as imaging of the brain of newborns through the skull. Compared to conventional imaging techniques such as X-ray and MRI, optical tomography has the following advantages: (a) it uses light, usually near infra-red light, to probe the medium, thus reducing the radiation harm; (b) unlike other methods which infer tumors from secondary effects (for example, X-ray observes calcification of blood vessels and MRI looks at water content of tissues), it is able to target tumor cells at the molecular level; (c) optical sources and sensors are well developed and inexpensive which reduces the cost for using it compared to other tomography modalities.

In traditional diffusion optical tomography (DOT), sinusoidally modulated, continuous wave or pulsed excitation light is launched at the subject where it will undergo absorption and scattering before it exits the subject. Detectors observe the exiting lights, and the observation can be used to reconstruct the map of absorption or scattering properties. Contrast of such properties can then indicate anomalies. For example, tumors usually have a larger blood supply compared to the surrounding tissues and different levels of blood associates with different levels of light absorption. See [10] for a comprehensive review on this method.

It has been noted that the contrast of optical properties between healthy and

tumor tissues can be relatively low and the use of fluorescent agents can improve the contrast. There are different ways that these agents can help with increasing contrast: some injected fluorophores may prefer to accumulate in diseased tissues because of larger blood flow around the diseased tissues; some may have different decay properties in diseased tissues than in healthy tissues, giving another way to localize the tumor. Moreover, some agents can selectively target receptors specific to cancer cells, which leads to a substantial improvement in the contrast. One of the immediate merits of such improvements is that it facilitates early diagnosis when the difference between healthy and diseased tissues are even smaller [52]. See [2, 3] for an overview of this topic.

In the following section, we will briefly describe the mathematical models that we use for the behavior of light. Simplifications (i.e., diffusion approximation, or, DA) are made under assumptions that are inherent in the media where optical tomography is used. Based on the DA model, we outline the system of equations—the forward model—for the FDOT. The reconstruction of optical properties is modeled under the Bayesian framework and we attempt to solve a three dimensional problem with real geometry. In order to give a reasonable resolution both the parameter and the state discretization are of high dimensions, which makes it difficult for any sampling scheme. Therefore, we utilize a deterministic method to a) locate the maximal a posteriori estimate which then serves as the starting point of our sampler, and b) adaptively refine both the parameter and state meshes. The use of adaptive finite element methods effectively reduces both the dimensionality of parameterization and the complexity of the model, thus ensuring a saving in computing time. We then apply our Markov chain preconditioned Monte Carlo scheme to sample the posterior distribution in parallel. Further analysis is conducted based on the generated samples.

5.1 The forward model

5.1.1 The radiative transfer equation and the diffusion approximation

The physics behind optical tomography is the propagation of light through a material medium which can be modeled through a conservation law that accounts for gains and losses of photons due to scattering and absorption. In particular, the light intensity obeys the radiative transport equation (RTE, also: *Boltzmann equation*):

$$\frac{1}{c}\frac{\partial I}{\partial t} + \hat{\mathbf{s}} \cdot \nabla I + \mu_a I + \mu_s I = \mu_s \int p(\hat{\mathbf{s}}, \hat{\mathbf{s}}') I(r, \hat{\mathbf{s}}') d\hat{\mathbf{s}}', \tag{5.1}$$

where c is the speed of light in the soft tissue, $r \in \mathbb{R}^n$ is the position vector, $\hat{s} \in \mathbb{S}^{n-1}$ is the direction vector, $I(r, \mathbf{s})$ is the *radiance* and $\mu_a \geq 0$ and $\mu_s \geq 0$ are the *absorption* and *scattering* coefficients, respectively. The kernel function $p(\hat{\mathbf{s}}, \hat{\mathbf{s}}')$ is the *scattering phase function* which is a probability density for the radiation to scatter from $\hat{\mathbf{s}}$ to $\hat{\mathbf{s}}'$. Therefore, the phase function satisfies

$$\int p(\hat{\mathbf{s}}, \hat{\mathbf{s}}') d\hat{\mathbf{s}}' = \int p(\hat{\mathbf{s}}, \hat{\mathbf{s}}') d\hat{\mathbf{s}} = 1. \tag{5.2}$$

The RTE is generally difficult to solve both analytically or numerically and the *diffusion approximation* (DA) is widely used to simplify the model. The DA is shown to be valid in strongly scattering media where the scattering length $l_s = 1/\mu_s$ is small compared to the distance of propagation (i.e., the *mean free path*). The standard method for the approximation is the P_1 approximation which first expands the angular dependence of the intensity and then truncates the expansion at the first order. Here we adopt another way, namely, the asymptotic expansion following [44]. This method takes higher orders of the spherical harmonics into account and shows

that they vanish in a natural way. Moreover, in contrast to the P_1 approximation which does not lead to boundary conditions, the asymptotic expansion yields the boundary conditions systematically.

For the DA to be valid, we have to assume that the scattering coefficient is large, the absorption coefficient is small, the domain is close to unbounded and the time is long enough. These assumptions enable us to use a multi-scale analysis by setting

$$\mu_s \to \frac{\mu_s}{\epsilon}, \quad \mu_a \to \epsilon\mu_a, \quad t \to \frac{t}{\epsilon},$$

after which (5.1) becomes

$$\frac{\epsilon^2}{c}\frac{\partial I}{\partial t} + \epsilon\hat{\mathbf{s}}\cdot\nabla I + \epsilon^2\mu_a I + \mu_s I = \mu_s \int p(\hat{\mathbf{s}},\hat{\mathbf{s}}')I(r,\hat{\mathbf{s}}')d\hat{\mathbf{s}}'. \tag{5.3}$$

Now we can use the asymptotic expansion on $I(r,\hat{\mathbf{s}}')$ to get

$$I(r,\hat{\mathbf{s}}') = \sum_{n=0}^{\infty} \epsilon^n I_n(r,\hat{\mathbf{s}}').$$

Insert this expression into (5.3), and compare the same ordered terms on both sides of the equation gives

$$\mu_s(1-K)I_n = \left(\mu_a + \frac{1}{c}\frac{\partial}{\partial t}\right) I_{n-2} + \hat{\mathbf{s}}\cdot\nabla I_{n-1}, \tag{5.4}$$

where we define $I_{-2}(r,\hat{\mathbf{s}}) = I_{-1}(r,\hat{\mathbf{s}}) = 0$, and the integral operator K is defined by

$$(Kf)(r,\hat{\mathbf{s}}) = \int p(\hat{\mathbf{s}},\hat{\mathbf{s}}')f(r,\hat{\mathbf{s}}')d\hat{\mathbf{s}}'.$$

Assume that the phase function is isotropic, namely, $p(\hat{\mathbf{s}},\hat{\mathbf{s}}') = p(\hat{\mathbf{s}}\cdot\hat{\mathbf{s}}')$, then we can expand $p(\hat{\mathbf{s}}\cdot\hat{\mathbf{s}}')$ as follows:

$$p(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}') = \sum_{l=0}^{\infty} \alpha_l(r) P_l(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}) = \sum_{l=0}^{\infty} \alpha_l(r) \sum_{m=-l}^{l} Y_{lm}(\hat{\mathbf{s}}) Y_{lm}^*(\hat{\mathbf{s}}')$$

where P_l are the Legendre polynomials, Y_{lm} are the spherical harmonics and the second equality is obtained using an addition theorem. Now we can render the integral operator as:

$$(Kf)(r, \hat{\mathbf{s}}) = \sum_{l=0}^{\infty} \alpha_l(r) \int P_l(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}) f(r, \hat{\mathbf{s}}') d\hat{\mathbf{s}}' \tag{5.5}$$

or,

$$(Kf)(r, \hat{\mathbf{s}}) = \sum_{l=0}^{\infty} \alpha_l(r) \sum_{m=-l}^{l} Y_{lm}(\hat{\mathbf{s}}) \int Y_{lm}^*(\hat{\mathbf{s}}') f(r, \hat{\mathbf{s}}') d\hat{\mathbf{s}}'. \tag{5.6}$$

From this equation, we can immediately tell that the $\alpha_l(r)$'s are the eigenvalues for the operator K and the corresponding eigenfunctions are $Y_{lm}(\hat{\mathbf{s}})$, $m = -l, \cdots, l$. Also, we will be using the following particular cases: when $l = 0$ we have

$$P_0(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}) = 1, \qquad \alpha_0(r) = 1$$

and when $l = 1$ we have

$$P_1(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}) = \hat{\mathbf{s}} \cdot \hat{\mathbf{s}}, \qquad \alpha_1(r) = \int p(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}') \hat{\mathbf{s}} \cdot \hat{\mathbf{s}}' d\hat{\mathbf{s}}'.$$

Now let us analyze cases for (5.4) at $n = 0, 1$ and 2. When $n = 0$, (5.4) becomes

$$\mu_s(1 - K) I_0 = 0.$$

Note that due to the unity condition (5.2) we deduce that $I_0(r, \hat{\mathbf{s}}) = I_0(r)$, that is, I_0 is constant in the angular direction. When $n = 1$, (5.4) gives that

$$\mu_s(1-K)I_1 = -\hat{\mathbf{s}}\cdot\nabla I_0,$$

from which it is easy to solve for I_1:

$$I_1 = -\frac{1}{\mu_s(1-\alpha_1(r))}\hat{\mathbf{s}}\cdot\nabla I_0.$$

Finally, we consider (5.4) for $n=2$, which gives

$$\begin{aligned}\mu_s(1-K)I_2 &= \left(\mu_a + \frac{1}{c}\frac{\partial}{\partial t}\right) I_0 + \hat{\mathbf{s}}\cdot\nabla I_1 \\ &= \left(\mu_a + \frac{1}{c}\frac{\partial}{\partial t}\right) I_0 + \hat{\mathbf{s}}\cdot\nabla\left(-\frac{1}{\mu_s(1-\alpha_1(r))}\hat{\mathbf{s}}\cdot\nabla I_0\right).\end{aligned}$$

The solvability of the above equation is that the right hand side integrates to zero over $\hat{\mathbf{s}}$. Thus, we end up with:

$$\frac{1}{c}\frac{\partial\Phi}{\partial t} = \nabla\cdot\left(\frac{1}{3\mu_s(1-\alpha_1(r))}\nabla\Phi(r,t)\right) - \mu_a\Phi(r,t), \tag{5.7}$$

where $\Phi(r,t) = \int I_0(r,\hat{\mathbf{s}})d\hat{\mathbf{s}}$ is the photon fluence field. This equation is the diffusion approximation for the RTE.

With an additional assumption that the source is time harmonic with modulation frequency ω, the diffusion approximation can also be formulated in the frequency domain as

$$-\nabla\cdot(D\nabla\Phi(r,\omega)) + (\mu_a + \frac{i\omega}{c})\Phi(r,\omega) = 0, \tag{5.8}$$

where we denote the coefficient in front of $\nabla\Phi(r,t)$ in (5.7) as D.

Note that the $\alpha_1(r)$ is the anisotropic scattering and it is zero when we consider only isotropic scattering. Note also that $l^* = 1/((1-\alpha_1(r))\mu_s)$ is the transport mean free path. In the P_N approximation method it is derived to be $l^* = 1/((1-\alpha_1(r))\mu_s+$

μ_a) and since $\mu_a = o(\mu_s)$ it is asymptotically equivalent to the derivation here.

5.1.2 Fluorescence enhanced optical tomography

In fluorescence optical tomography, light at the fluorophore's excitation wavelength is introduced to the subject. This gives rise to an excitation field which we conveniently denote as u. When the excited fluorophore decays to its ground state, it re-emits light at a longer wavelength or, equivalently, at a lower energy level. In our notation, the emitted light field is denoted as v. Using the diffusion approximation (5.8), the photon propagation in the FDOT can be expressed as:

$$-\nabla \cdot [D_x \nabla u] + k_x u = 0, \qquad in\ \Omega, \tag{5.9}$$

$$-\nabla \cdot [D_m \nabla v] + k_m v = \beta_{xm} u, \quad in\ \Omega, \tag{5.10}$$

where x stands for excitation and m stands for emission, $u,\ v \in H^1(\Omega)$ are complex functions that describe the photon fluence fields , and the coefficients satisfy the following equations:

$$D_* = \frac{1}{3(\mu_{a*i} + \mu_{a*f} + \mu_s')}, \qquad k_* = \frac{i\omega}{c} + \mu_{a*i} + \mu_{a*f}, \qquad \beta_{xm} = \frac{\phi \mu_{axf}}{1 - i\omega\tau},$$

where $*$ could be either the excitation x or the emission m, μ_{a*i} corresponds to the absorption coefficient for the endogenous chromophores, μ_{a*f} corresponds to the absorption coefficient for the exogenous fluorophore, and μ_s' is the scattering coefficient reduced by the anisotropic factor $1 - \alpha_1(r)$ as above. For boundary conditions, we incorporate the Robin-type condition to model the NIR excitation source:

$$2D_x \frac{\partial u}{\partial n} + \gamma u + S = 0, \qquad 2D_m \frac{\partial v}{\partial n} + \gamma v = 0 \quad on\ \partial\Omega, \tag{5.11}$$

where $S = S(\mathbf{r}), \mathbf{r} \in \partial\Omega$ is the spatially variable excitation boundary source, n denotes the outward normal to the surface and γ is an optical property modeling the impedance mismatch between air and tissue.

5.2 The Bayesian model of the inverse problem

We now build the posterior distribution under the Bayesian framework. Typically, the parameters of interest in fluorescent tomography problems are the absorption coefficient $\mu_{axf}(r)$ and the decay coefficient $\tau(\mathbf{r})$ where $\mathbf{r} \in \Omega$. In our discussion, however, we will only focus on the absorption coefficient and assume all other parameters are already known. Therefore, the objective of the inverse problem is to estimate a posterior distribution of the absorption coefficient

$$q = \mu_{axf}$$

given observations of the excitation light field. In our numerical setting, the field $v(\mathbf{r})$ is measured at N locations on the top surface Γ of the object, and we denote the reference measurements as $\mathbf{z} = \{z_i, i = 1, 2, \cdots, N\}$. We impose independent and additive noise model for the data, that is, we assume

$$z_i = v_i(q) + \epsilon,$$

where $v_i(x)$ is the pointwise evaluation of the excited light $\mathbf{v}$, the solution of equations (5.9), (5.10) and (5.11), at the same measurement location as that of z_i, and ϵ is a white noise with standard deviation σ_d. Therefore, the unnormalized likelihood function can be written as

$$\pi(z|q) \propto \exp\left(-\frac{1}{2}\frac{\sum_{i=1}^{N}|v_i(q) - z_i|^2}{\sigma_d^2}\right).$$

We also assume a Gaussian prior for the parameter field, namely,

$$\pi(q) \propto \exp\left(-\frac{1}{2}\frac{||q||_\Omega^2}{\sigma_m^2}\right),$$

where $||\cdot||_\Omega$ represents the L_2 norm over the three dimensional domain of interest. Multiplying the prior distribution with the likelihood function then gives us the unnormalized posterior distribution:

$$\pi(q|z) \propto \exp\left(-\frac{\sum_{i=1}^{N}|v_i(q)-z_i|^2}{2\sigma_d^2} - \frac{||q||_\Omega^2}{2\sigma_m^2}\right). \tag{5.12}$$

5.3 Deterministic inversion and adaptive mesh refinement

If we jumped to sampling the posterior distribution immediately, we would have to face the following difficulties. First of all, we would not know where a good starting point is. Initiating the Markov chain from a point far from the region of interest implies a long burn-in period, and it becomes especially intractable in the current three dimensional problem where each sample evaluation is extremely expensive. Another problem is that we do not have knowledge about the mesh discretization for either the parameter field or the state variables to begin with. A fine enough mesh is naturally preferred for better reconstruction, but it adds more computational burden. In our three dimensional problem, for a structured mesh, every refinement results in eight times more cells. Several global refinements will soon render the problem computationally intractable.

To mitigate the problems above, we adopt a strategy that first computes the maximum a posterior (MAP) estimator using deterministic inversion. This will allow us to obtain the MAP estimator as a starting point for the subsequent sampling process. If the posterior distribution were a multivariate normal distribution, for

example, the MAP would coincide with the conditional mean (CM) estimation, and hence we could expect an efficient sampling by starting from this point. Even though our problem is not multivariate normal due to the nonlinearity of the forward model, we can still expect to capture the significant features of the posterior distribution much faster by starting from an optimal point. In addition, we use adaptive mesh refinement [5, 8, 9], for the parameter and state meshes respectively, along the iterations that solve the deterministic inverse problems. This provides a discretization that is efficient to evaluate while maintaining accuracy. More importantly, we naturally obtain both the approximation model and the full order model by extracting two refinements along this iterative refinement process. We will show in the following that MCPMC samples efficiently using this setup even without inserting the approximation error model discussed in Section 4.2.

5.3.1 Deterministic inversion

Modeling the deterministic problem: In deriving the algorithm for the deterministic inversion, we follow the exposition previously given in [5, 8]. The MAP point is computed through solving the following optimization problem:

$$\min_q J(q) = \frac{1}{2}\sum_{i=1}^{N} |v_i(q) - z_i|^2 + \frac{\beta}{2}||q||_\Omega^2, \tag{5.13}$$

where $J(q) = (-\log \pi(q|z))$, $\beta = \sigma_d^2/\sigma_m^2$, and $\mathbf{v} = \{v_1, v_2, \cdots, v_N\}$ satisfy a weak variational form of the forward equations (5.9)–(5.11) that is defined as:

$$\begin{aligned} A(q; [u\ v])([\phi\ \psi]) := (D_x \nabla u, \nabla \phi)_\Omega + (k_x u, \phi)_\Omega + \frac{\gamma}{2}(u, \phi)_{\partial\Omega} + \frac{1}{2}(S, \phi)_{\partial\Omega} \\ + (D_m \nabla v, \nabla \psi)_\Omega + (k_m v, \psi)_\Omega + \frac{\gamma}{2}(v, \psi)_{\partial\Omega} - (\beta_{xm} u, \psi)_\Omega = 0, \end{aligned} \tag{5.14}$$

where ϕ and ψ are any function from the test space H^1, the Sobolev space of functions having square integrable first order derivatives and parentheses $(\cdot,\cdot)_X$ denote the inner product in the $L^2(X)$ sense. Notice that, with the Bayesian modeling of an additive noise and a Gaussian prior, the MAP problem (5.13) is exactly the same as the Tikhonov regularization problem in Section 1.2.

Now, the Lagrangian function of this constrained optimization problem becomes:

$$L(q;[u\ v];[\lambda^u\ \lambda^v]) = J(q;v) + A(q;[u\ v])([\lambda_u\ \lambda_v]), \tag{5.15}$$

where λ_u and λ_v are the Lagrange multipliers for the equality constraints. If we denote $\theta = (u, v, q, \lambda^u, \lambda^v)$, we know that the minimizer will have to satisfy the optimality condition:

$$L_\theta(\theta)(\theta') = 0, \qquad \forall\ \theta' = (u', v', q', {\lambda_u}', {\lambda_v}'), \tag{5.16}$$

which, in its partial derivative forms, is written as

$$\begin{aligned}
L_u(\theta)(u') &= (D_x\nabla u', \nabla\lambda^u)_\Omega + (K_x u', \lambda^u)_\Omega + \frac{\gamma}{2}(u', \lambda^u)_{\partial\Omega} - (\beta_{xm}u', \lambda^v)_\Omega = 0,\\
L_v(\theta)(v') &= (v - z, v')_\Gamma + (D_m\nabla v', \nabla\lambda^v)_\Omega + (K_m v', \lambda^v)_\Omega + \frac{\gamma}{2}(v', \lambda^v)_{\partial\Omega} = 0,\\
L_{\lambda^u}(\theta)(\lambda^{u\prime}) &= (D_x\nabla u, \nabla\lambda^{u\prime})_\Omega + (K_x u, \lambda^{u\prime})_\Omega + \frac{\gamma}{2}(u, \lambda^{u\prime})_{\partial\Omega} + \frac{1}{2}(S, \lambda^{u\prime})_{\partial\Omega} = 0,\\
L_{\lambda^v}(\theta)(\lambda^{v\prime}) &= (D_m\nabla v, \nabla\lambda^{v\prime})_\Omega + (K_m v, \lambda^{v\prime})_\Omega + \frac{\gamma}{2}(v, \lambda^{v\prime})_{\partial\Omega} - (\beta_{xm}u, \lambda^{v\prime})_\Omega = 0,\\
L_q(\theta)(q') &= \beta(q, q')_\Omega + (D_{x,q}(q')\nabla u, \nabla\lambda^u)_\Omega + (K_{x,q}(q')u, \lambda^u)_\Omega\\
&\quad + (D_{m,q}(q')\nabla v, \nabla\lambda^v)_\Omega + (K_{m,q}(q')v, \lambda^v)_\Omega - (\beta_{xm,q}(q')u, \lambda^v)_\Omega = 0.
\end{aligned}$$

Gauss-Newton-CG method: To solve this set of nonlinear equations, we adopt a Gauss-Newton-CG method which is an iterative method. The Gauss-Newton method

is a modification of Newton's method which, at each iteration k, seeks an update direction $d\theta = (du, dv, dq, d\lambda^u, d\lambda^v)$ by solving the equation

$$L_{\theta\theta}(\theta^k)(d\theta, \theta') = -L_\theta(\theta^k).$$

Note that as the Lagrange multipliers are proportional to the misfit $||v - z||^2_\Gamma$ they should be negligible near the optimal point. Hence we simplify the above second order derivative by neglecting all the terms that contain either λ^u or λ^v. This gives the set of partial differential equations for the Gauss-Newton method

$$\begin{aligned}
&(D^k_x \nabla u', \nabla d\lambda^u)_\Omega + (K^k_x u', d\lambda^u)_\Omega + \frac{\gamma}{2}(u', d\lambda^u)_{\partial\Omega} - (\beta^k_{xm} u', d\lambda^v)_\Omega && = - L_u(\theta^k)(u'),\\
&(dv, v')_\Gamma + (D_m \nabla v', \nabla d\lambda^v)_\Omega + (K_m v', d\lambda^v)_\Omega + \frac{\gamma}{2}(v', d\lambda^v)_{\partial\Omega} && = - L_v(\theta^k)(v'),\\
&(D^k_x \nabla du, \nabla \lambda^{u\prime})_\Omega + (K^k_x du, \lambda^u)_\Omega + \frac{\gamma}{2}(du, \lambda^{u\prime})_{\partial\Omega}\\
&\qquad + (D^k_{x,q}(dq)\nabla u, \nabla \lambda^{u\prime})_\Omega + (K^k_{x,q}(dq)u, \lambda^{u\prime})_\Omega && = - L_{\lambda^u}(\theta^k)(\lambda^{u\prime}),\\
&-(\beta^k_{xm} du, \lambda^{u\prime})_\Omega + (D_m \nabla dv, \nabla \lambda^{v\prime})_\Omega + (K_m dv, \lambda^{v\prime})_\Omega +\\
&\quad \frac{\gamma}{2}(dv, \lambda^{v\prime})_{\partial\Omega} + (D_{m,q}(dq)\nabla v, \nabla \lambda^{v\prime})_\Omega + (K_{m,q}(dq)v, \lambda^{v\prime})_\Omega && = - L_{\lambda^v}(\theta^k)(\lambda^{v\prime}),\\
&\beta(dq, q')_\Omega + (D^k_{x,q}(q')\nabla u, \nabla d\lambda^u)_\Omega + (K^k_{x,q}(q')u, d\lambda^u)_\Omega\\
&\quad + (D_{m,q}(q')\nabla v, \nabla d\lambda^v)_\Omega + (K_{m,q}(q')v, d\lambda^v)_\Omega - (\beta_{xm,q}(q')u, d\lambda^v)_\Omega && = - L_q(\theta^k)(q'),
\end{aligned}$$

for any vector $(u', v', q', \lambda^{u\prime}, \lambda^{v\prime})$ in the product of test spaces.

Discretization and solution: To solve this coupled system of equations numerically, we use the finite element method: for variables $u, v, \lambda^u, \lambda^v$, we discretize the domain with a mesh $\{\mathbb{T}^s_k\}$ and with continuous elements; for the parameter field q, since we expect to detect localized objects (e.g., the tumor), we implement a different mesh $\{T^q_k\}$ with a discontinuous finite element [5]. While it is natural to have the

mesh as fine as possible so that the numerical solution meets enough accuracy, a too fine parameter mesh will however lead to a high dimensional inference problem. The separate mesh scheme manages the parameter and state variable discretization separately, hence avoiding such a dilemma. With such a discretization, and if we group the unknowns to $dp = [du\ dv]^T$ and $d\lambda = [d\lambda^u\ d\lambda^v]^T$, we can express the above Gauss-Newton equations in the matrix form

$$\begin{bmatrix} M & 0 & P^T \\ 0 & R & C^T \\ P & C & 0 \end{bmatrix} \begin{bmatrix} dp_k \\ dq_k \\ d\lambda_k \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}. \tag{5.17}$$

When the discretization is fine, the linear system (5.17) can be very large, reaching millions of unknowns. Even worse, the block matrix on the left of equation is usually indefinite, making the solution process slower by excluding iterative methods such as the conjugate gradient (CG) methods. Therefore, we adopt an approach based on the Schur complement. That is, we first use block elimination to obtain the *reduced KKT system*:

$$\begin{aligned} \{R + C^T P^{-T} M P^{-1} C\} dq_k &= F_2 - C^T P^{-T} F_1 + C^T P^{-T} M P^{-1} F_3, \\ P dp_k &= F_3 - C dq_k, \\ P^T d\lambda_k &= F_1 - M dp_k. \end{aligned}$$

Now, in this reduced system, the Schur complement matrix $R + C^T P^{-T} M P^{-1} C$ is generally symmetric and positive definite, thus it can be solved using the CG method. More importantly, the size of the Schur complement is only the same as the parameter discretization, which is a far smaller than the original block system.

5.3.2 Adaptive mesh refinement

We adaptively refine both the parameter and the state meshes respectively along the Gauss-Newton-CG iterations. This has at least two benefits. First, we are able to start from a coarser mesh for the iterations at the beginning where accurate search direction is not demanded, thus saving substantial computation time. Second, the adaptive refinements lead to a reasonable parameterization: the parameter mesh is only refined where the parameter is rough. This way, we mitigate the curse of dimensionality that could happen later in the sampling without losing the accuracy of inference. We implement different refinement criteria for the state and parameter mesh refinements, respectively. The state mesh is refined when either (a) the norm of the residual $L_\theta(\theta_k)$ is reduced by a preset factor from the first iteration on the same mesh or (b) the line search radius is below a preset threshold. Condition (a) is an indicator that the iterations have made sufficient progresses and can explore finer structures, while condition (b) occurs when the iterations get trapped in a local region and one mesh refinement may help get rid of the trap. Refer to [8] for more elaboration on this strategy.

5.4 Stochastic inversion using MCPMC

In this section, we present numerical results for a three dimensional problem. The domain is contained in the rectangular box of size 8cm $\times$ 8cm $\times$ 4cm with an irregular top surface obtained from an actual measurement of an animal tissue (i.e., a pig groin in this case). A single target model is assumed where the target is set to be a sphere about 2cm down the top surface and the sphere has a radius of about 0.5cm. We illuminate the top surface with a ring shaped diffractive light pattern.

We generate synthetic data using a very fine discretization of 211,152 cells and 1,014,572 degrees of freedom. This mesh is finer than any other mesh we adopt for the

reconstruction, thus avoiding the inverse crime. The reference measurements z_i are then taken from 366 locations on the top surface. See Figure 5.1 for a demonstration of the geometry, the illumination light and the mesh discretization for computing the synthetic data.

We take the data noise σ_d to be 3% of the maximal magnitude of all the measurements, and set σ_m to be such that $\sigma_m^2 = \sigma_d^2/\beta$, where the regularization weight β is computed during the Gauss-Newton iterations, following the method described in [8].

We use the code developed in [8] for the deterministic inversion. The code is run for 8 Gauss-Newton iterations, which results in a parameter mesh with 529 cells. Given that we use a piecewise constant element for the parameter field, the discretized parameter field also has a dimension of 529. Similarly, we obtain a coarse state mesh with 5,415 cells and 30,544 unknowns. With three additional adaptive refinements on the coarse mesh, we obtain the fine level mesh with 58,524 cells and 301,880 unknowns. The parameter, coarse state and fine state meshes are shown in Figure 5.2. Also, we obtain the MAP estimator at this iteration of the Gauss-Newton method. This MAP vector is then taken as the initial sample for any sampling we will be conducting.

One constraint inherent to this inverse problem is that as a physical parameter μ_{axf} should be real and positive. We hence use a log-normal proposal for the samplers in this experiment. That is, suppose the current sample is q_k, we propose a new sample q^* through

$$q^* \sim \text{Log–}\mathcal{N}(\ln q_k, \Sigma)$$

for some covariance matrix Σ. In implementations, we can first propose a new vector $\eta \sim \mathcal{N}(\ln(q_k), \Sigma)$ and then let $q^* = \exp(\eta)$. This step guarantees that proposed

samples have real and positive components, while the symmetry of log-normal distributions makes sure that formula for computing the acceptance ratio remain the same as random walk samplers.

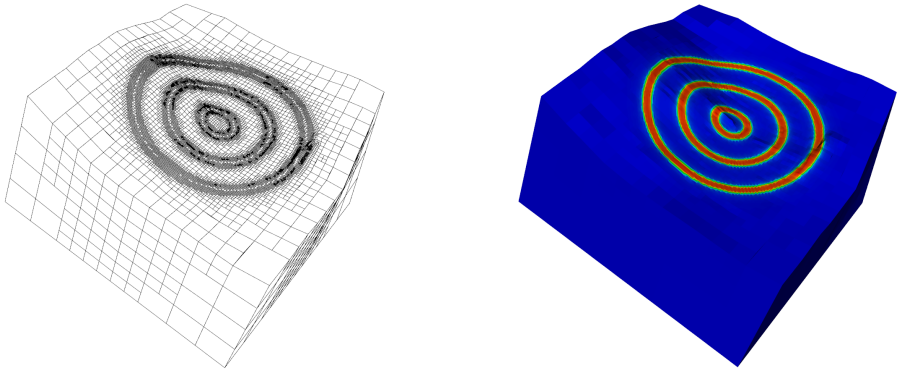


Figure 5.1: Left: Mesh that is used for generating the synthetic data. Right: Diffractive excitation light that illuminates the tissue.

After testing for several hundred samples on both the coarse and fine mesh, we obtain that the average run time for each fine evaluation is 41.50 seconds, while the averaged evaluation time on the coarse mesh is 3.20 seconds. We use a total of 50 processors to run the MCPMC in parallel. Two of these processors are used for the master processor and the preconditioned coarse chain respectively, and the other 48 processors are used as slave processors that repeatedly evaluate likelihood functions on the fine mesh. We tune both the preconditioning MH chain and the perturbations so that they have acceptance ratios of $20\% - 30\%$, which leads us to believe that these MH updates reach good enough statistical efficiency according to [57].

In the actual experiment, we observe that the acceptance ratio for the updating

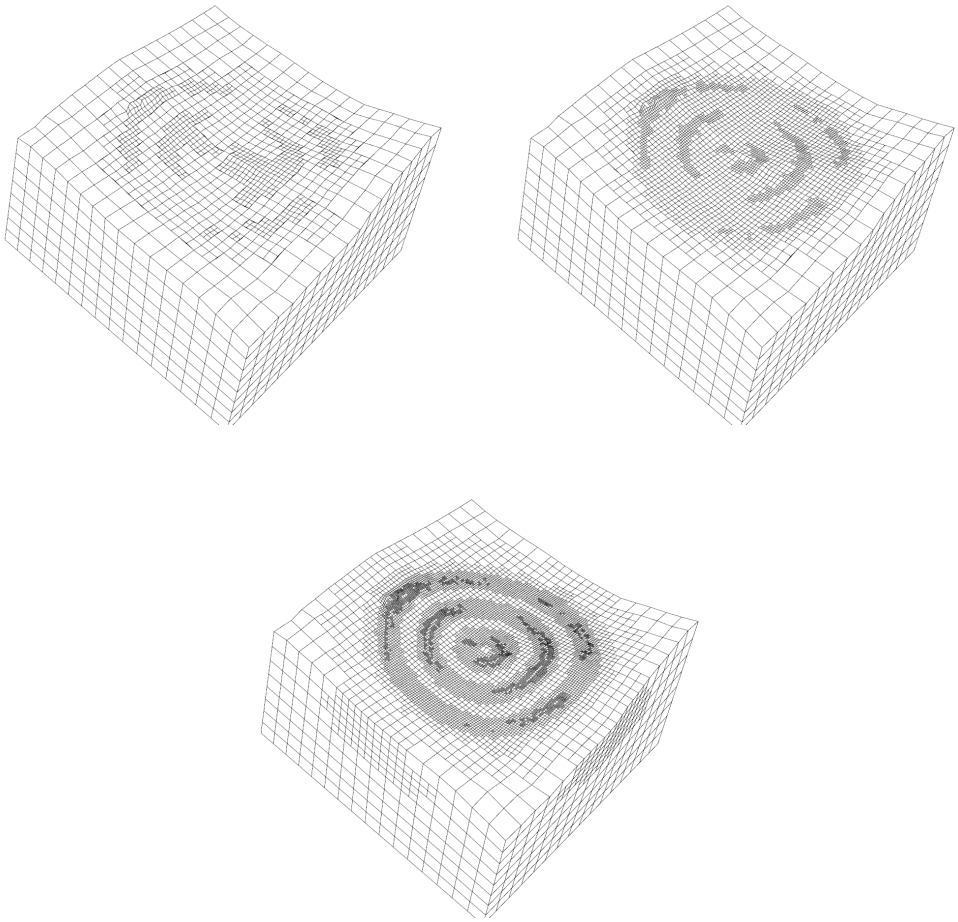


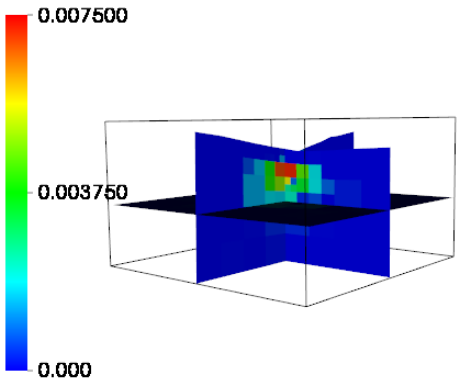
Figure 5.2: Meshes used in the MCPMC sampling: the top left figure shows the parameter field discretization, the top right figure shows the coarse grid for state variables and the bottom figure shows the fine grid for state variables.

step is about 22.6%, namely, nearly a quarter of the random selections from the pool of coarse level Markov chain samples get accepted as new seeds for the fine level perturbations. While accepting enough new candidates guarantees the variability of the sampler, rejecting a certain amount of those who come from the coarse level distribution means that the sampler is effectively transforming from the approximate model to the true model.

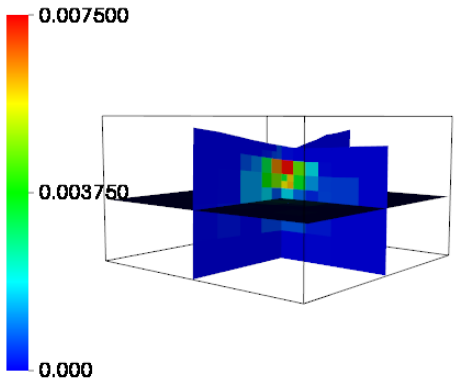
To evaluate the quality of the samples, we let the fine MH chain run for about 30 days and obtain 60,000 samples, out of which we discard the first 20,000 samples as the burn-in period and use the remaining samples to compute all statistics. At the same time, we take the same number of samples from the MCPMC. Since the MCPMC is tuned so that the speed of producing fine level samples is about the same as the speed of producing coarse samples, the sampling process for getting these many MCPMC samples only needs about 3 days. In Figure 5.3, cross-section plots of first order statistic (i.e., sample mean) are presented. It is clear that the mean estimate from both samplers provides satisfying reconstructions that accurately capture the location of the target in the $x - y$ plane and stay close to the real target in the z direction. The overestimate of the target in the z-direction is not uncommon, as is explained in [38] and the references therein. As an additional comparison, we also plot the reconstruction from the MAP estimator to demonstrate that MCPMC has much more similarity to the fine MH chain and has diverged from the MAP estimator, which was chosen as the starting point for both samplers.

In Figure 5.4, we show the significant region of the second order statistic (i.e. sample standard deviation) from both samplers, where the threshold for defining the significant region is chosen to be 50% of the largest standard deviation. Both the MCPMC and fine MH indicate the same region that has a large uncertainty.

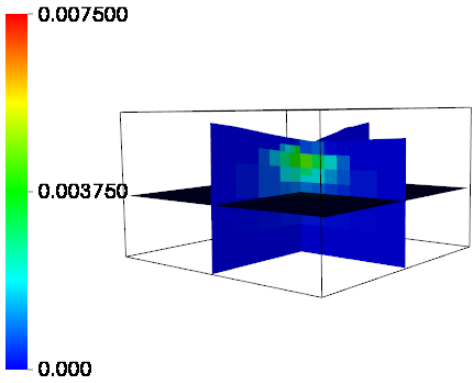
We conduct some local analysis in Figure 5.5, where we plot histograms of eight



(a) CM estimator on with fine MH.



(b) CM estimator with MCPMC.



(c) MAP estimator.

Figure 5.3: A cross section of the object showing the conditional mean (CM) estimates with coarse chain MH, fine chain MH and MCPMC sampling, also in comparison with the MAP estimate computed from the deterministic inversion.

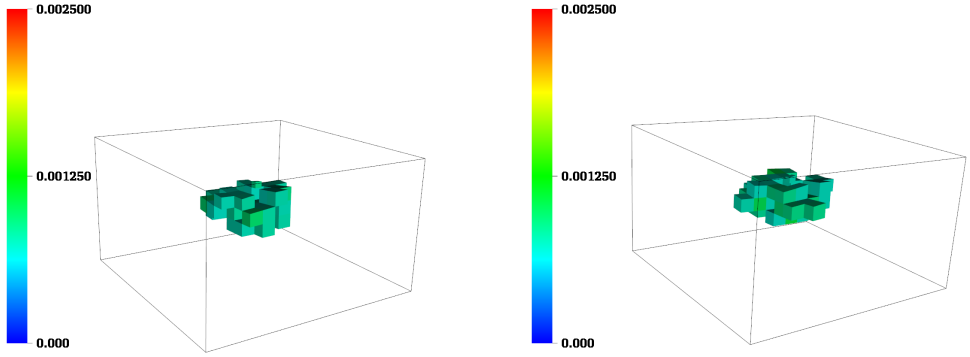


Figure 5.4: Comparison of the range of cells whose standard deviation is above a certain threshold (0.00125). Left: Standard deviation using samples from the MH sampler on the fine grid. Right: Standard deviation using samples from the MCPMC sampler.

parameter components that correspond to the 'target' pixels, namely, the pixels whose magnitude are above 60% of the largest magnitude. For most of these components, the two histograms overlap very well, indicating that the MCPMC is at least providing similar reconstruction and uncertainty quantification. This intuitive observation is further evidenced by the numeric comparison listed in Table 5.1: all eight but one of the components have a mean estimation error under a quarter of their corresponding standard deviation. The only outlier is q_{528} which has a significant discrepancy in both histograms and a larger mean estimation error. Noting that the standard deviation from both samplers for this pixel is relatively large, we can argue that this component might have a larger uncertainty directly from the Bayesian model rather than as a result of the sampling process. In addition, the limited number of samples may indicate a not-yet-convergent situation for this particular pixel. Even in cases like this, we can see the benefits of using MCPMC which

is a much faster sampler that we will be able to generate many more samples in a short time to test if good convergence has been reached.

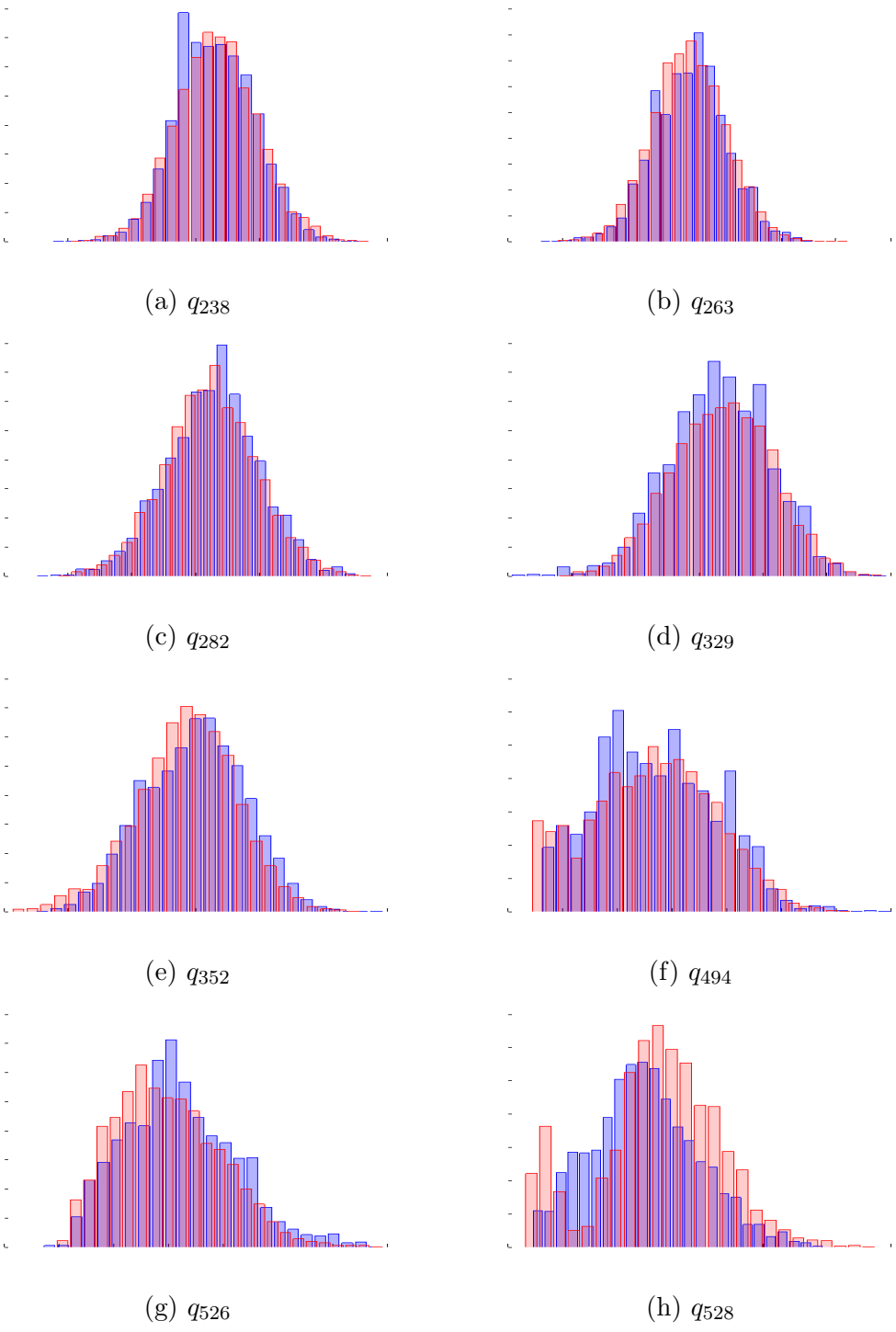
To quantitatively compare the statistical efficiency of both samplers we compute two numeric indicators for both samplers. The first one is the integrated autocorrelation time (IACT) τ of the parameter field $\mathbf{q}$ following the definition in (2.5). An efficient sampler tends to generate samples that are less correlated with other samples in the process, and hence should demonstrate a small IACT. In practice, it is not possible to compute the infinite sum and thus we truncate it at lag $t = 3000$ and choose the maximal τ as a representation.

The second indicator is the mean squared jump (MSJ) distance [49] that is generally defined, for samples $(\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N)$ as

$$\text{MSJ} := \frac{1}{N} \sum_{i=0}^{N-1} ||\mathbf{q_{i+1}} - \mathbf{q_i}||^2.$$

For a given sampler, a larger MSJ value usually means a faster mixing rate and hence a better convergence.

We randomly select four components to compute these two indicators, that is, we define the function $h(\mathbf{q}) := q_{i_0}$ where i_0 are four random indices. We report the computed indicators in Table 5.2. We also provide the plots that compare the autocorrelation functions for different samplers in Figure 5.6. In all these randomly chosen parameters, it is definitive that MCPMC has a smaller IACT, a larger MSJ and an autocorrelation curve that drops to zero much faster. These results suggest that MCPMC can also provide convergence gains in the statistical sense, in addition to the fact that it produces samples much faster with multiple processors. In example (3.2.1), we analyze the speedup of this tomography experiment in a quantitative manner. The speedup is defined by combining the acceleration of sample generation



(a) q_{238} (b) q_{263}

(c) q_{282} (d) q_{329}

(e) q_{352} (f) q_{494}

(g) q_{526} (h) q_{528}

Figure 5.5: Comparison of histograms at pixels which have estimated magnitude greater than 60% of the maximal magnitude—these are considered as the target pixels. The histogram from MCPMC is plotted in blue and that from the fine MH is plotted in red.

Component	mean(MCPMC)	mean(fine MH)	stddev(MCPMC)	stddev(fine MH)	relative error
q_{238}	0.733	0.730	0.0656	0.0624	4.8%
q_{263}	0.729	0.730	0.0670	0.0660	1.5%
q_{282}	0.737	0.721	0.0743	0.0708	22.6%
q_{329}	0.526	0.525	0.0755	0.0808	1.2%
q_{352}	0.499	0.484	0.0874	0.0775	19.3%
q_{494}	0.513	0.523	0.244	0.226	4.4%
q_{526}	0.586	0.590	0.214	0.190	2.1%
q_{528}	0.438	0.585	0.245	0.240	61.5%

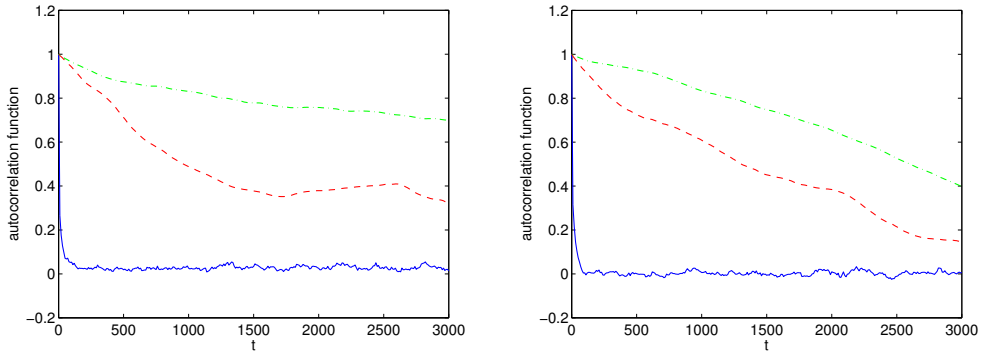
Table 5.1: Comparison of conditional mean (CM) and standard deviation (stddev) estimation between MCPMC and fine MH samplers at pixels which have estimated magnitude greater than 60% of the maximal magnitude—these are considered as the target pixels. We also compute a relative error defined as $(\text{mean}_{\text{MCPMC}} - \text{mean}_{\text{fine-MH}})$ $/\text{stddev}_{\text{fine-MH}}$, i.e., we compute the difference between the mean estimates from both samplers normalized by the standard deviation of the fine MH sampler.

Parameter	IACT(f)	IACT(m)	MSJ(f)	MSJ(m)
q_1	150.86	10.56	1.35×10^{-14}	1.09×10^{-9}
q_{150}	147.17	3.19	1.09×10^{-10}	3.05×10^{-9}
q_{261}	169.85	3.33	1.81×10^{-40}	1.51×10^{-34}
q_{494}	64.86	10.02	1.96×10^{-8}	2.13×10^{-6}

Table 5.2: The integrated autocorrelation time (IACT) and the mean square jump (MSJ) computed for several components of the parameter vector. In the table, fine MH is abbreviated as "f" and MCPMC is abbreviated as "m".

and the improvement of statistical efficiency. From the analysis therein, we can conclude that the data here demonstrate a superlinear speedup.

To summarize, these numerical results in this section indicate that for a realistic and complex application, the MCPMC algorithm produces samples that not only have similar or better statistical properties than a standard Metropolis-Hastings (MH) sampler, but can be produced at least ten times faster than with the MH process using the 50 processors we have allocated for this experiment.



(a) q_1 (b) q_{150}

(c) q_{261} (d) q_{494}

Figure 5.6: Autocorrelation function of randomly selected parameter components plotted for autocorrelation time up to 3000. In each plot, the red curve is for the fine MH sampler, green for the coarse MH sampler and blue for the MCPMC sampler.

6. CONCLUSIONS

Bayesian methods provide a sound framework for modeling inverse problems. The posterior distribution built with such methods can not only provide estimates of parameters but also the uncertainty of these estimations. However, it requires additional techniques to extract information from the posterior distribution. Markov chain Monte Carlo (MCMC) is a category of well-studied algorithms that can be used for sampling such distributions, but since they require evaluating the forward model repeatedly, they can become very slow for large scale inverse problems. In addition, their sequential nature make them difficult to be parallelized. We construct a new framework of algorithms, namely, the Markov chain preconditioned Monte Carlo (MCPMC) methods, for efficiently sampling posterior distributions with parallel computing. Through a Markov chain running an approximate model which is usually much faster to compute, we then are able to randomly select candidates from this chain for further processing with the target model. While these additional steps can be computationally expensive, they can be performed at the same time on multiple processors. This gives enormous saving of time because previous MCMC methods can only run sequentially on a single processor even if we have tens of thousands of processors available. In addition, through an example of two dimensional mixture Gaussian distribution—a typical multi-modal example—we demonstrate that the new sampler can also enhance the statistical efficiency through properly chosen approximate distributions.

For an MCPMC to run efficiently, it is important to build an approximate distribution that is both close to the target posterior distribution and is fast to sample. We demonstrate two viable techniques to achieve this through a two dimensional

elliptic and a three dimensional optical tomography problem. In the experiment of the elliptic PDE inverse problem, we adopt an enhanced error model to estimate the discrepancy between the approximation and the target; in the optical tomography problem, we utilize a hierarchy of adaptive finite elements where the finer mesh is used for the target distribution and the coarser for the approximation. Numerical results show that both techniques can be applied to the framework of MCPMC, and that the sample estimates from the new sampler match that from a traditional MH algorithm.

It is worth pointing out that the MCPMC provides an algorithmic structure that is compatible with any existing MCMC method. Both the sampling of the approximation model and the perturbations for further correction can take advantage of any advanced MCMC algorithm for better efficiency. One possibility is to explore adaptivity of the sampler. A vast literature has contributed to adaptive MCMC methods where the sampler learns to locate the next sample more and more efficiently by summarizing the sample history. In MCPMC, we have both the approximate and the accurate models which will give more information about where we should sample. We also have both the base chain simulation and the perturbation steps that can benefit from such adaptivity. Whether to use statistics from the approximate model or the accurate model to direct either of the aforementioned MCMC steps is currently still unclear. Also, as the samples are evaluated on both models, it is straightforward to compute their differences. These differences can improve the modeling of the enhanced error model discussed in Section 4.2. This additional step costs nearly nothing, but it should refine the error model effectively as the number of sample evaluations increase. In general, it will be an important topic to investigate the possibility of building the approximate model along the sampling process intelligently. Information obtained from more and more samples may help us better understand

which features in the model are of significant importance to narrow the gap between the approximation and the target distributions. Such understanding may then guide us to build approximate models that are closer to the target distribution or faster to sample, either of which can further increase the efficiency of MCPMC.

REFERENCES

[1] S. Arridge, J. Kaipio, V. Kolehmainen, M. Schweiger, E. Somersalo, T. Tarvainen, and M. Vauhkonen. Approximation errors and model reduction with an application in optical diffusion tomography. *Inverse Problems*, 22:175–195, 2006.

[2] S. R. Arridge. Optical tomography in medical imaging. *Inverse problems*, 15(2):R41, 1999.

[3] S. R. Arridge and J. C. Schotland. Optical tomography: forward and inverse problems. *Inverse Problems*, 25(12):123010, 2009.

[4] Y. Atchadé, G. Fort, E. Moulines, and P. Priouret. Adaptive markov chain monte carlo: theory and methods. *Preprint*, 2009.

[5] W. Bangerth. A framework for the adaptive finite element solution of large inverse problems. *SIAM Journal on Scientific Computing*, 30:2965–2989, 2008.

[6] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II — a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4), 2007.

[7] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier, B. Turcksin, and T. D. Young. The deal.II library, version 8.0. *arXiv preprint http://arxiv.org/abs/1312.2266v3*, 2013.

[8] W. Bangerth and A. Joshi. Adaptive finite element methods for the solution of inverse problems in optical tomography. *Inverse Problems*, 24(3):034011, 2008.

[9] W. Bangerth and R. Rannacher. *Adaptive finite element methods for differential equations*. Springer Basel AG, Basel, 2003.

[10] D. A. Boas, D. H. Brooks, E. L. Miller, C. A. DiMarzio, M. Kilmer, R. J. Gaudette, and Q. Zhang. Imaging the body with diffuse optical tomography. *Signal Processing Magazine, IEEE*, 18(6):57–75, 2001.

[11] T. Bodin, M. Sambridge, H. Tkalčić, P. Arroucau, K. Gallagher, and N. Rawlinson. Transdimensional inversion of receiver functions and surface wave dispersion. *Journal of Geophysical Research: Solid Earth (1978–2012)*, 117(B2), 2012.

[12] S. C. Brenner and R. Scott. *The mathematical theory of finite element methods*, volume 15. Springer, New York, 2008.

[13] A. Brockwell. Parallel markov chain monte carlo simulation by pre-fetching. *Journal of Computational and Graphical Statistics*, 15(1):246–261, 2006.

[14] A. E. Brockwell, P. Del Moral, and A. Doucet. Sequentially interacting Markov chain Monte Carlo methods. *Annals of Statistics*, 38(6):3387–3411, 2010.

[15] S. Brooks and A. Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4):434–455, 1998.

[16] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC Press, Boca Raton, 2011.

[17] G. Casella and R. Berger. *Statistical Inference*. Duxbury Press, Pacific Grove, 2001.

[18] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM review*, 41(1):85–101, 1999.

[19] M. K. Cowles and B. P. Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

[20] T. Cui, C. Fox, and M. O'Sullivan. Adaptive error modelling in MCMC sampling for large scale inverse problems. *Report, Univeristy of Auckland, Faculty of Engineering, Auckland*, 2011.

[21] S. Dadi, R. Gibson, and K. Wang. Velocity log upscaling based on reversible jump MCMC simulated annealing. *in preparation*.

[22] P. Del Moral, A. Doucet, and J. A. Sequential Monte Carlo for Bayesian computation. *Bayesian Statistics*, 8:1–34, 2007.

[23] P. Del Moral, A. Doucet, and A. Jarsa. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society, Series B.*, 68(3):411–436, 2006.

[24] D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

[25] Y. Efendiev, T. Hou, and W. Luo. Preconditioning Markov chain Monte Carlo simulations using coarse-scale models. *SIAM Journal of Scientific Computing*, 28(2):776–803, 2006.

[26] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Springer, New York, 2004.

[27] A. Gelfand and S. A. F. M. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409, 1990.

[28] A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992.

[29] C. J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 7(4):473–483, 1992.

[30] A. Grama. *Introduction to parallel computing*. Pearson Education, Harlow, 2003.

[31] P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[32] P. J. Green and D. I. Hastie. Reversible jump MCMC. *Genetics*, 155(3):1391–1403, 2009.

[33] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, Cambridge, 1999.

[34] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, 2006.

[35] H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, 7(2):223–242, 2001.

[36] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[37] B. Jin. A variational Bayesian method to inverse problems with impulsive noise. *Journal of Computational Physics*, 231(2):423–435, 2012.

[38] A. Joshi, W. Bangerth, K. Hwang, J. C. Rasmussen, and E. M. Sevick-Muraca. Fully adaptive FEM based fluorescence optical tomography from time-dependent measurements with area illumination and detection. *Medical Physics*, 33(5):1299–1310, 2006.

[39] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Springer, New York, 2004.

[40] M. C. Kennedy and A. O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[41] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[42] V. Kolehmainen, M. Schweiger, I. Nissilä, T. Tarvainen, S. R. Arridge, and J. P. Kaipio. Approximation errors and model reduction in three-dimensional diffuse optical tomography. *JOSA A*, 26(10):2257–2268, 2009.

[43] P.-S. Koutsourelakis. A multi-resolution, non-parametric, Bayesian framework for identification of spatially-varying model parameters. *Journal of computational physics*, 228(17):6184–6211, 2009.

[44] E. W. Larsen and J. B. Keller. Asymptotic solution of neutron transport problems for small mean free paths. *Journal of Mathematical Physics*, 15:75–81, 1974.

[45] F. Liang, C. Liu, and R. Carroll. *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*. Wiley, Chichester, 2010.

[46] F. Liang and W. H. Wong. Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. *Journal of the American Statistical Association*, 96(454):653–666, 2001.

[47] J. Liu, F. Liang, and W. Wong. The multiple-try method and local optimization in Metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000.

[48] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, New York, 2008.

[49] J. Martin, L. C. Wilcox, C. Burstedde, and O. Ghattas. A stochastic Newton MCMC method for large-scale statistical inverse problems with application to seismic inversion. *SIAM Journal on Scientific Computing*, 34(3):A1460–A1487, 2012.

[50] K. L. Mengersen and R. L. Tweedie. Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24(1):101–121, 1996.

[51] N. Metropolis, A. Rosenblutn, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Technical report, Los Alamos Scientific Lab.; Chicago Univ., 1953.

[52] A. B. Milstein, S. Oh, K. J. Webb, C. A. Bouman, Q. Zhang, D. A. Boas, R. Millane, et al. Fluorescence optical diffusion tomography. *Applied Optics*, 42(16):3081–3094, 2003.

[53] J. Nocedal and S. Wright. *Numerical optimization: Springer series in operations research and financial engineering*. Springer, New York, 2006.

[54] D. S. Oliver, A. C. Reynolds, and N. Liu. *Inverse theory for petroleum reservoir characterization and history matching*. Cambridge University Press, Cambridge, 2008.

[55] C. Robert and G. Casella. A short history of Markov chain Monte Carlo: subjective recollections from incomplete data. *Statistical Science*, 26(1):102–115, 2011.

[56] G. O. Roberts and J. S. Rosenthal. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability*, 44:458–475, 2007.

[57] G. O. Roberts, J. S. Rosenthal, et al. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical science*, 16(4):351–367, 2001.

[58] M. K. Sen and P. L. Stoffa. *Global optimization methods in geophysical inversion*. Elsevier, Amsterdam, 1995.

[59] M. Snir. *MPI–the Complete Reference: The MPI core*, volume 1. MIT press, Cambridge, 1998.

[60] A. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[61] A. Tarantola. *Inverse problem theory and methods for model parameter estimation*. SIAM, Philadelphia, 2005.

[62] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.

[63] J. Wang and N. Zabaras. Hierarchical bayesian models for inverse problems in heat conduction. *Inverse Problems*, 21(1):183, 2005.

APPENDIX A

REVERSIBLE JUMP MARKOV CHAIN MONTE CARLO METHOD

In this appendix, we discuss a generalized version of the Makrov chain Monte Carlo (MCMC)—the reversible jump Markov chain Monte Carlo (RJMCMC) algorithm. It is now drawing attention from the inverse problem community and the author of the thesis has also conducted some research using this method.

In all the aforementioned MCMC algorithms, there is one embedded hypothesis: the dimension of all the samples is *known* and it is fixed throughout the sampling process. In the meanwhile, there are Bayesian modeling techniques which pursue more generality—they allow the dimension of the parameters to also be a parameter! One of such instances is seen in nonparametric modeling of the Bayesian problem. For example, in [43], the parameter field is modeled as a weighted sum of different kernels, and the number of kernels to use is unspecified in the modeling.

Formally, in variational dimension modeling, instead of considering a single parameter space, we consider a sequence of spaces $\mathcal{X}_k$ indexed by k. The complete parameter space is

$$\mathcal{X} = \bigcup_k \{k\} \times \mathcal{X}_k,$$

on which the augmented posterior distribution is defined as

$$\pi(x, k) = \pi(x)\rho(k),$$

where $\rho(k)$ is some prior distribution for the number of parameters. To sample the augmented distribution, [31] developed the reversible jump algorithm which takes care of the jump between different dimension spaces automatically during the sam-

pling process.

The main feature of the reversible jump algorithm is that when it jumps between spaces of different dimensions, it retains the reversibility of the sampler by supplementing the dimensionality difference with artificial spaces. Specifically, suppose $k_1 < k_2$, the current state is x_{k_1}, and one wish to jump to a state of dimension k_2, one can establish a *dimension match* mechanism by proposing a matching vector $u \sim q_{k_1 \to k_2}(\cdot)$ such that the new vector (x_{k_1}, u) is of dimension k_2. This augmented vector is then mapped to a new state

$$x_{k_2} = g(x_{k_1}, u)$$

through a one-to-one and differentiable mapping $g : \mathcal{X}_{k_1} \times \mathcal{X}_{k_2 - k_1} \to \mathcal{X}_{k_2}$. Once the new state x_{k_2} is proposed, a Metropolis-Hastings type acceptance probability is computed as

$$\alpha([k_1\ x_{k_1}], [k_2\ x_{k_2}]) = 1 \wedge \frac{\pi(x_{k_2}) q(k_2 \to k_1)}{\pi(x_{k_1}) q(k_1 \to k_2) q_{k_1 \to k_2}(u)} \left| \frac{\partial_g(x_{k_1}, u)}{\partial(x_{k_1}, u)} \right|,$$

where $q(k_i \to k_j)$ is the probability of jumping from one dimension to another, and the last fraction is the Jacobian for the map $g(x_{k_1}, u)$ which maintains the right volume of distributions in different dimensions. To maintain reversibility, the reversible step from $\mathcal{X}_{k_2}$ to $\mathcal{X}_{k_1}$ is then made deterministic and the acceptance rate for the reverse proposal is always set to be

$$\alpha([k_2\ x_{k_2}], [k_1\ x_{k_1}]) = \alpha([k_1\ x_{k_1}], [k_2\ x_{k_2}])^{-1}.$$

It is noticeable that when the parameter dimension is fixed, namely, $k_1 \equiv k_2$, the aforementioned formula naturally become the Metropolis-Hastings algorithm. In that

sense, the reversible jump algorithm is a more generalized algorithm. For detailed overview of this algorithm, see the survey [32] or chapter 3 in [16].

The reversible jump MCMC (RJMCMC) is a powerful tool for transdimensional inverse problems where it is beneficial to make the dimension of parameters flexible and let the sampler to decide proper dimensions. To illustrate, we give an example from [21] where transdimensional inversion and a reversible jump MCMC based simulated annealing is applied to a well log upscaling problem. Well log is a detailed record of geological formations obtained at a borehole. Once a hole is drilled, it is almost possible to obtain measurements from any depth that is above the deepest location of the hole. Therefore, well logs are always of high frequency. In contrast, surface seismic data are often of low frequency and hence it is of necessity to upscale the well logs when comparison with the seismic data is conducted. Typical upscaling techniques divide the domain of full depth into smaller intervals and run averaging methods such as the Backus average over each interval. One of the difficulties here is to determine the number as well as the displacements of the boundaries for these small intervals. In this example, both the number of layers and their depths are modeled as a part of the parameters and a reversible jump algorithm with birth-depth move is used to infer these parameters.

In Figure A.1, upscaling results for a well log from using fixed dimensional MH algorithm and RJMCMC are shown. The two plots on the left show the histogram of layer boundary locations and the upscaled well log from MCMC sampling with a fixed number of layers, only allowing layer boundaries to move, whereas the two plots on the right give the corresponding results from RJMCMC sampling which allows inserting and deleting layers in addition. It is clear that with the flexibility to also vary the number of layers, the sampler is able to add additional layers to places where better resolution is demanded. As a consequence, the upscaled result returns

better accuracy by capturing more details especially in greater depths.

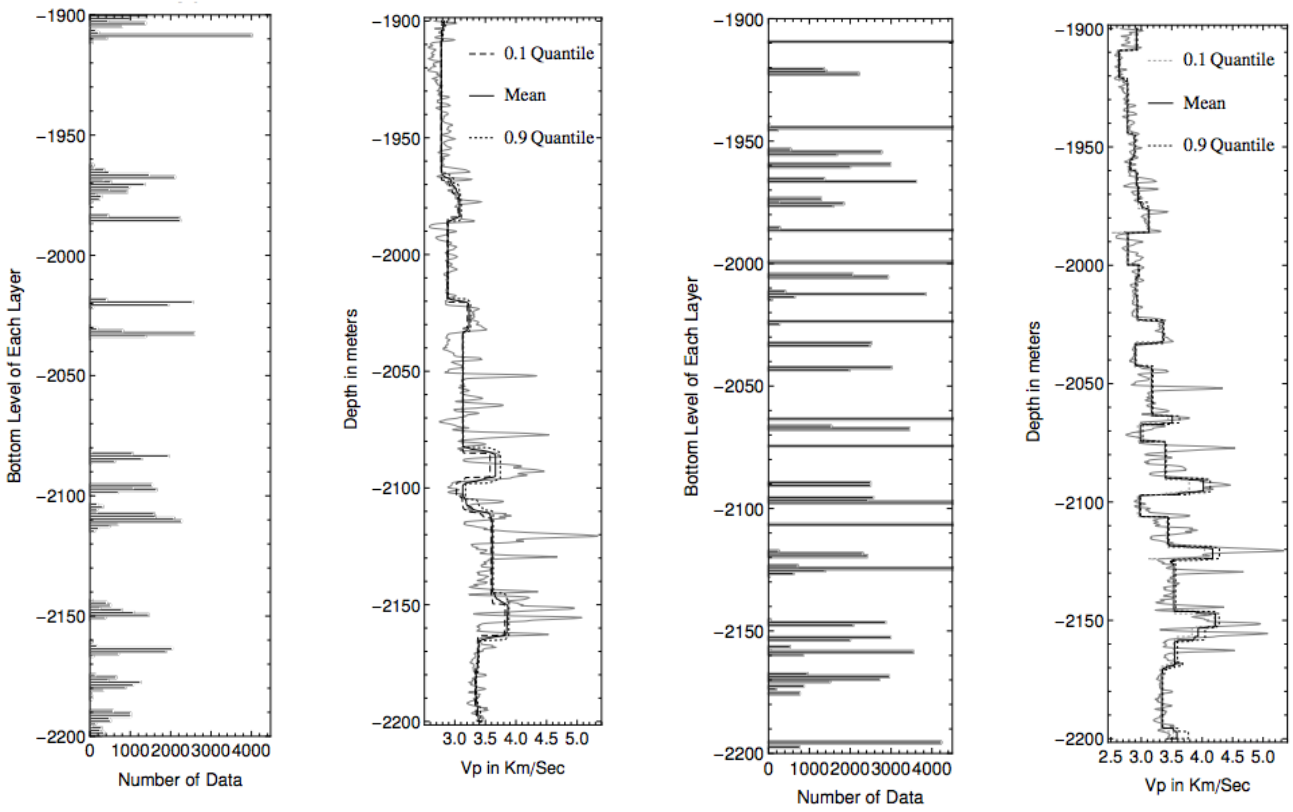


Figure A.1: Upscaled results for a sample well log: the left two plots are the histogram of the boundary locations and the estimated upscaling with a fixed number of layers; the right two plots are the histogram of the boundary locations and the estimated upscaling with variable number of layers and reversible jump MCMC. In the upscaled comparison plots, the wiggly solid line is the true well log, and the smoother lines are the mean, 10% and 90% upscaled estimates as indicated in the plot. Plot obtained from [21].