LAPPEENRANTA UNIVERSITY OF TECHNOLOGY
Department of Information Technology
Laboratory of Applied Mathematics

Antti Solonen

Monte Carlo Methods in Parameter Estimation of Nonlinear Models

The topic of this Master's thesis was approved by the department council of the Department of Information Technology on 15 November 2006.

The examiners of the thesis were Professor Heikki Haario and PhD Tuomo Kauranne. The thesis was supervised by Professor Heikki Haario.

Lappeenranta, December 7, 2006

Antti Solonen
Tervahaudankatu 1 A 17
53850 Lappeenranta
+358 400 734378
antti.solonen@lut.fi

ABSTRACT

Lappeenranta University of Technology
Department of Information Technology

Antti Solonen

Monte Carlo Methods in Parameter Estimation of Nonlinear Models

Master's Thesis

2006

112 pages, 40 figures, 4 tables and 1 appendix

Examiners: Professor Heikki Haario
PhD Tuomo Kauranne

Keywords: Monte Carlo, MCMC, Predictive Inference, Population Monte Carlo, Chemical Kinetics

One of the main tasks in statistical analysis of mathematical models is the estimation of the unknown parameters in the models. In this thesis we are interested, instead of single estimates, in the distributions of the unknown parameters and numerical methods suitable for forming them, especially in cases where the model is nonlinear with respect to the parameters.

From different numerical methods the Markov Chain Monte Carlo (MCMC) methods are especially emphasized. These computationally intensive methods have become popular in statistical analysis during the last decades, mainly due to increased computational power. The theory of both Markov Chains and Monte Carlo simulations is presented to the extent that is needed in justifying the MCMC methods. From the recently developed methods especially the adaptive MCMC methods are discussed. The approach of the thesis is practical and thus different issues related to the implementation of the MCMC methods are emphasized.

The empirical part of the work consists of five example models that are studied using the methods discussed in the theoretical part. The models describe chemical reactions and are given as ordinary differential equation systems. The models are collected from chemists in Lappeenranta University of Technology (Lappeenranta, Finland) and Åbo Akademi (Turku, Finland).

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tietotekniikan osasto

Antti Solonen

Monte Carlo -menetelmät epälineaaristen mallien parametrien estimoinnissa

Diplomityö

2006

112 sivua, 40 kuvaa, 4 taulukkoa ja 1 liite

Tarkastajat: Professori Heikki Haario
FT Tuomo Kauranne

Hakusanat: Monte Carlo, MCMC, Prediktiiviset jakaumat, Populaatio Monte Carlo, Kemiallinen kinetiikka

Keywords: Monte Carlo, MCMC, Predictive Inference, Population Monte Carlo, Chemical Kinetics

Yksi keskeisimmistä tehtävistä matemaattisten mallien tilastollisessa analyysissä on mallien tuntemattomien parametrien estimointi. Tässä diplomityössä ollaan kiinnostuneita tuntemattomien parametrien jakaumista ja niiden muodostamiseen sopivista numeerisista menetelmistä, etenkin tapauksissa, joissa malli on epälineaarinen parametrien suhteen.

Erilaisten numeeristen menetelmien osalta pääpaino on Markovin ketju Monte Carlo -menetelmissä (MCMC). Nämä laskentaintensiiviset menetelmät ovat viime aikoina kasvattaneet suosiotaan lähinnä kasvaneen laskentatehon vuoksi. Sekä Markovin ketjujen että Monte Carlo -simuloinnin teoriaa on esitelty työssä siinä määrin, että menetelmien toimivuus saadaan perusteltua. Viime aikoina kehitetyistä menetelmistä tarkastellaan etenkin adaptiivisia MCMC menetelmiä. Työn lähestymistapa on käytännönläheinen ja erilaisia MCMC -menetelmien toteutukseen liittyviä asioita korostetaan.

Työn empiirisessä osuudessa tarkastellaan viiden esimerkkimallin tuntemattomien parametrien jakaumaa käyttäen hyväksi teoriaosassa esitettyjä menetelmiä. Mallit kuvaavat kemiallisia reaktioita ja kuvataan tavallisina differentiaaliyhtälöryhminä. Mallit on kerätty kemisteiltä Lappeenrannan teknillisestä yliopistosta ja Åbo Akademista, Turusta.

PREFACE

This work is done as part of the Finnish Center of Excellence for Inverse Problems Research. The work is funded by the Academy of Finland. I want to thank the Academy of Finland and the laboratory of applied mathematics at Lappeenranta University of Technology for making this learning experience possible.

I wish to thank the supervisor of the thesis, Professor Heikki Haario, for giving valuable comments and guidance, and Tuomo Kauranne for examining the thesis. In addition, acknowledgment belongs to Marko Laine from Helsinki University for helping in technical issues. From the laboratory of applied mathematics at Lappeenranta University of technology I especially thank Tapio Leppälampi for cooperation and brainstorming. Timo Haakana, Arto Laari and Markku Kuosa from the department of chemical engineering (LUT), and Johan Wärnå from the Process Chemistry Centre (Åbo Akademi), deserve acknowledgment for providing data and models for the empirical part of the thesis.

I thank my family for support and financial security throughout the studying period - and before. I am grateful also to my friends, especially among Piimäkassi and AIESEC Saimaa, for making my studies at Lappeenranta unforgettable. I especially thank Maiju Kansanen for making sure that the difference between working hours and leisure time has been statistically significant.

ALKUSANAT

Tämä diplomityö on tehty osana Suomen Akatemian Inversio-ongelmien huippuyksikköohjelmaa. Haluan kiittää Suomen Akatemiaa ja Lappeenrannan Teknillisen Yliopiston sovelletun matematiikan laitosta tämän oppimiskokemuksen mahdollistamisesta.

Haluan kiittää työn ohjaajaa, Professori Heikki Haariota, arvokkaista kommenteista ja ohjauksesta, sekä Tuomo Kaurannetta työn tarkastamisesta. Myös Marko Laineelle Helsingin Yliopistosta kuuluu kiitos teknisissä asioissa auttamisesta. Lappeenrannan teknillisen yliopiston sovelletun matematiikan laitokselta erityiskiitokset Tapio Leppälammelle yhteistyöstä ja aivoriihistä. Kiitän Timo Haakanaa, Arto Laaria ja Markku Kuosaa LTY:n kemiantekniikan osastolta, sekä Johan Wärnåta Process Chemistry Centeristä (Åbo Akademi) datan ja mallien työstämisestä diplomityön empiiriseen osaan.

Kiitän perhettäni tuesta ja taloudellisen turvallisuuden takaamisesta koko opiskeluajan - ja sitä ennenkin. Kiitokset myös ystäville, etenkin Piimäkassille ja AIESEC Saimaan jäsenille, joiden ansiosta opiskeluaika Lappeenrannassa ei tule koskaan unohtumaan. Kiitän erityisesti Maiju Kansasta, jonka ansiosta työtuntien ja vapaa-ajan välinen ero on ollut tilastollisesti merkitsevä.

7. joulukuuta 2006

Contents

VOCABULARY

MC	Monte Carlo
MCMC	Markov Chain Monte Carlo
MAP	Maximum a Posteriori
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimate
iid	Independent and Identically Distributed (random variables)
LLN	Laws of Large Numbers
CLT	Central Limit Theorem
PDF	Probability Density Function
CDF	Cumulative Density Function
MH	Metropolis-Hastings Algorithm
SC	Single Component Metropolis-Hastings Algorithm
LSQ	Least Squares
MSE	Mean Square Error
RSS	Residual Sum of Squares
AP	Adaptive Proposal Algorithm
AM	Adaptive Metropolis Algorithm
SCAM	Single Component Adaptive Metropolis Algorithm
DR	Delayed Rejection Method
DRAM	Delayed Rejection Adaptive Metropolis Algorithm
PMC	Population Monte Carlo
GMM	Gaussian Mixture Models
ODE	Ordinary Differential Equation
SMC	Sequential Monte Carlo
SIS	Sequential Importance Sampling
SIR	Sequential Importance Resampling
EM	Expectation-Maximization Algorithm

NOTATIONS

General

θ	Unknown parameters
$\mathbf{y}$	Measurements
ϵ	Measurement errors
X	Desing matrix
$f(\mathbf{x};\theta)$	Model with known $\mathbf{x}$ and unknown parameters θ
$p(\theta;\mathbf{y})$	Joint probability distribution of θ and $\mathbf{y}$
$p(\mathbf{y}\|\theta)$	Likelihood
$\pi_{pr}(\theta)$	Prior
$\pi(\theta\|\mathbf{y})$	Posterior
p_ϵ	PDF for error ϵ
$E_{p(x)}(f(x))$	Expectation of $f(x)$ under $p(x)$
μ	Expectation
σ^2	Variance
s^2	Sample variance
C	Covariance
$P("prop")$	Probability that $proposition$ holds
$\overline{X}$	Sample mean of vector X
SS_θ	Sum of squares with parameters θ
J	Jacobian matrix
H	Hessian matrix

Distributions

$U(a,b)$	Uniform distribution in the interval (a,b)
$N(\mu,C)$	Normal distribution with mean μ and covariance C

$\log -N(\mu, C)$	Log-Normal distribution with mean μ and covariance C
Inv-$\chi^2(n, S^2)$	Scaled inverse χ^2 distribution with parameters n and S^2
$\Gamma^{-1}(a, b)$	Inverse Gamma distribution with parameters a and b

Monte Carlo Methods

$\hat{I}$	Monte Carlo estimate for integral I
$q(x)$	Envelope function in Accept-Reject method
$g(x)$	Importance function (distribution) in importance sampling
w	Importance weights

Markov Chains

$\lambda(s_i)$	Distribution for the initial state s_i of a Markov Process
π^*	Stationary distribution of a Markov Process
P	Transition probability matrix
P_n	Transition probability matrix for n steps

MCMC and PMC

$q(.\|\theta)$	Proposal distribution at point θ
$R(k)$	k:th order autocorrelation
$d(x)$	Kernel density estimate at x
$K(x)$	Kernel function at x
iq	Interquartile range of samples

1 Introduction

During the last few decades there has been a revolution in applying methods based on random sampling to problems of statistical analysis. This is mainly due to increased computational power. In practice this development has led to increased applicability of Monte Carlo (MC) methodology. Especially Markov Chain Monte Carlo (MCMC) methods have become applicable and widely used in statistical analysis of mathematical models.

This work discusses the methods meant for an important task in statistical analysis of mathematical models - the estimation of unknown parameters and their distributions in models, based on measured data. The case of parameter estimation in *nonlinear* models is especially under investigation. In this case, some numerical methods have to be applied - in this work we concentrate on Monte Carlo methods that are based on random sampling. In addition, the most important theoretical and practical concepts behind the methods are discussed, in order to give the reader a comprehensive, yet practical view of the methods.

Thus, the problem in the work is not merely to give fixed, in some sense optimal values for the unknown parameters in the models, but to estimate also the *distribution* of the parameters. This leads to a Bayesian problem formulation where the unknown quantities in the models are thought to be random variables with certain distributions.

In this section the outline of the thesis is shortly presented, together with a description of the research approach and methodology. The most important research questions are presented in an explicit way to give a clear and concrete image of the goals of the thesis.

1.1 Outline

The thesis begins by introducing the basic idea of the Bayesian framework, together with the Bayes' rule (chapter 2). The basic theories behind Monte Carlo sampling methods are discussed in chapter 3 and the usage of MC methodology is justified. In particular, some MC sampling methods are introduced, including Accept-Reject methods and variance reduction methods such as importance sampling. In chapter 4, the essential theories about Markov Chains are presented to the extent that is needed to explain how the MCMC methods work. In addition, the basic MCMC methods are explained together with issues related to the implementation of the algorithms (chapter 5). The visualization of the output of the methods is especially taken into consideration.

Besides the basic MCMC methods based on the Metropolis and Gibbs algorithms, some new, recently developed versions of the algorithms are discussed in chapter 6. In particular, different adaptive MCMC methods are presented, together with conclusions about how they should be used and in which cases they are most useful. In addition, an alternative MC method, the Population Monte Carlo (PMC), is discussed in chapter 7. The foundation of the method, importance sampling, is presented in chapter 3.

Some development ideas and directions for future research are discussed in chapter 8 in a subjective manner. The ideas presented are based on observations made when working with the different methods.

The empirical part of the thesis (chapter 9) consists of five example models gathered from research groups in Lappeenranta University of Technology and Åbo Akademi. The presented methods are applied to these problems that are currently under research. The goal is on one hand to produce new information related to the models, such as uncertainties, parameter identifiability and correlations. On the other hand the work attempts to spread the methodology and tools to the researches utilizing statistical analysis in modeling. The case examples concentrate on different chemical reaction models and parameter estimation in them.

1.2 Approach and Methodology

The goal of the thesis is to provide the reader with a clear, yet practical view about some numerical methods available for evaluating the distribution of unknown parameters in nonlinear models. The bases of the methods are investigated theoretically, but the main emphasis of the work is on algorithms and implementation issues. The goal is that the reader is able to implement the methods after reading this thesis. That is, a major part of the proofs and theoretical background of the methods leans on references to literature sources. The strongly applying approach is chosen because of the lack of literature related to implementing the algorithms. The algorithms are presented in pseudo-code and platform-specific details are not addressed.

The work is done in a comparative manner. That is, one algorithm (Metropolis-Hastings) is considered to be the "standard" procedure, against which other methods are mirrored and to which new modifications are presented. On the other hand, two different fundamental approaches to the problem are presented: the MCMC methodology based on

Metropolis-type accept-reject rules and methods based on iterative importance sampling procedures.

The thesis is also a case-study - the methods are applied to one type of real life applications; ordinary differential equation systems describing chemical reactions. The goal is to show that the algorithms work in real applications and that they are relatively easy to implement. The collaboration with the chemical engineering groups is intended to lead to more wide application of the methods.

In practice the work is done by building simple implementations for the methods using the *MATLAB* software. In addition, a numerical software called *Modest* is used to produce the final output related to the case example models.

1.3 Research Questions

First of all, after the problem is formulated, it is essential to know how it can be expressed mathematically. This problem is handled in the work (section 2) through the Bayesian framework, the use of which has to be justified as well. As concrete research questions:

- *How can the distribution of the unknown parameters of a nonlinear model be represented mathematically?*
  - *Why is the Bayesian approach chosen and how does it differ from the classical Frequentist framework?*

As mentioned, the emphasis of the work is in the MCMC methods. The description of these requires, however, some basic concepts related to both Monte Carlo and Markov Chain theory. Thus, answers for the following questions are sought in chapters 3 and 4:

- *What do we mean by Monte Carlo methods and why do they produce correct results?*
- *How can the MCMC methods be justified from the Markov Chain point of view?*

The main research question is related to the MCMC methods themselves (chapter 5). The practical approach of the work is expressed by the following set of questions:

- *How can we form the distribution of the unknown parameters in nonlinear models using MCMC methods?*
  - *What kind of methods exist for the task?*
  - *What do we have to consider when we want to implement the methods?*
  - *How can we efficiently illustrate the results visually?*
  - *What kind of random sampling methods do we need to implement the MCMC methods? How do these work?*

An important part of the thesis is the demonstration of some of the recently developed versions of the MCMC methods and alternatives to them (chapters 6 and 7). In addition, one of the goals of the work is to come up with some ideas to improve the methods (chapter 8). These raise the following questions:

- *What kind of improvements have been introduced lately to the standard MCMC algorithms?*
  - *What kind of cases are these algorithms suitable for?*
- *How does Population Monte Carlo (PMC) work and how does it differ from MCMC?*
- *How could we develop the methods further?*

The purpose of the empirical part is to put the methods into action and apply the methods to a certain set of specific problems. That is, we ask

- *How can the methods be applied to ODE models?*
- *What kind of new information related to the example models do the methods produce?*

2 Bayesian Inference in Parameter Estimation

The general form of a nonlinear model is presented in equation (2.1). The model consists of measurements $\mathbf{y}$, known quantities $\mathbf{x}$ (constants, control variables etc.), unknown parameters θ and measurement error ϵ:

$$\mathbf{y} = f(\mathbf{x}; \theta) + \epsilon. \tag{2.1}$$

The problem is to estimate the unknown parameters θ based on the measurements $\mathbf{y}$ ([1], [2]). In this work a group of numerical methods for solving the problem, based on random sampling, are presented in the framework of Bayesian theory. That is, the error and the unknown parameters in the model are random variables and have a distribution - they are not thought to have a single "correct" value, but different possible values, others being more probable than the others.

2.1 Linear vs. Nonlinear Models

The model is said to be linear with respect to its unknown parameters θ if it can be written as $\mathbf{y} = f(X)\theta$, where the matrix X includes the design (input) variables. For linear models, exact analytic formulas exist for creating statistics (estimates and approximations about their precision) about the unknown parameters (see Appendix 1 for the basic formulas in linear case).

For nonlinear models, no exact theory exists for estimates of unknown parameters and for their distribution and accuracy. That is, numerical methods are needed in both finding the best estimate (nonlinear optimization task) and evaluating the distribution of the parameters. The emphasis of the work is on estimating the distribution of parameters of *nonlinear* models.

2.2 Bayesian vs. Frequentist Framework

In statistical analysis there are two major approaches to inference - the Frequentist and the Bayesian approach. In general, the goal in statistical inference is to make conclusions about a phenomenon based on observed data. In the Frequentist framework the observations made in the past are analyzed with a created model and the result is regarded

as confidence about the state of the real world. That is, we assume that the phenomenon modeled has statistical stability: the probabilities are defined as frequencies with which an event occurs if the experiment is run many times. An event with probability p is thought to occur pn times if the experiment is repeated n times.

In the Bayesian approach the interpretation of probability is subjective. The belief quantified before is updated to present belief through new observed data. In the Bayesian framework the probability is never just a frequency (single value), but a distribution of possible values. In the previous example the frequency pn can have different values of which other are more probable than the others - for every claim a probability can be assigned that tells how strong our belief about the claim is. That is, the Bayesian inference is based on assigning degrees of beliefs for different events.

A common task in statistical analysis is the estimation of the unknown model parameters. The Frequentist approach relies on estimators derived from different data sets (experiments) and a specific sampling distribution of the estimators. In the Bayesian approach the solution encompasses various possible parameter values. Therefore, the Bayesian approach is by nature suitable for modeling uncertainty in the model parameters and model predictions.

The Bayesian approach is based on *prior* and *likelihood* distributions of parameters. The prior distribution includes our beliefs about the problem beforehand, whereas the likelihood represents the probabilities of observing a certain set of parameter values. The prior and the likelihood are updated to a posterior distribution, which represents the actual parameter distribution conditioned on the observed data, through the Bayesian rule (section 2.3). [3], [4], [5]

2.3 Bayes' Rule

As stated above, the Bayesian solution to the parameter estimation task is the posterior distribution of the parameters, which is the conditional probability distribution of the unknown parameters given the observed data. That is, we are interested in the distribution with probability density function $\pi(\theta|\mathbf{y})$ where θ denotes the unknown parameter values and $\mathbf{y}$ contains the observations.

To define $\pi(\theta|\mathbf{y})$ we assume that there is a joint probability density function $p(\theta; \mathbf{y})$ that gives the probability for every combination of parameters and data. In the Bayesian frame-

work this function is expressed as

$$p(\theta; \mathbf{y}) = p(\mathbf{y}|\theta)\pi_{pr}(\theta), \tag{2.2}$$

where $\pi_{pr}(\theta)$ is the prior distribution that describes our prior knowledge of the parameters. Here $p(\mathbf{y}|\theta)$ is the likelihood function that gives the probability for receiving data $\mathbf{y}$ if we have parameter value θ. In order to receive the posterior probability density function the joint probability has to be normalized so that the probabilities sum to value 1. This scaling factor is the density function of all possible measurements, $p_Y(\mathbf{y})$. The posterior density can now be written in a form of the Bayesian Rule ([6], [1]):

$$\pi(\theta|y) = \frac{p(y|\theta)\pi_{pr}(\theta)}{p_Y(y)} \tag{2.3}$$

which is analogous to the Bayesian rule from the elementary probability calculus for two random variables A and B:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}. \tag{2.4}$$

The scaling factor (the marginal density of observations) can be calculated as the sum (integral) over all possible joint probabilities. That is, the Bayesian formula can be expressed with

$$\pi(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)\pi_{pr}(\theta)}{\int_{\mathbb{R}^d} p(\mathbf{y}|\theta)\pi_{pr}(\theta)d\theta}. \tag{2.5}$$

A simple analytical example of parameter estimation in the Bayesian framework is presented in chapter 2.3.4.

The tricky part in implementing Bayesian inference in practice is the normalizing constant that requires integration over an often high-dimensional space. This integral is seldom possible to calculate analytically. Deterministic methods based on the discretization of the space may not be feasible because of large computational complexity due to high dimension. This problem can be tackled, for example, with Monte Carlo integration methods (see chapter 3) or with Markov Chain Monte Carlo methods (see chapters 4-6) in which the need for computing these difficult integrals vanishes.

Before moving into Monte Carlo integration and MCMC methods in parameter estimation, we take a closer look on the role of prior and likelihood distributions from the point of view of parameter estimation.

2.3.1 Prior Distributions

As mentioned, the prior distribution describes our previous (a priori) knowledge about the unknown parameters in the model. With properly selecting the prior distribution we can emphasize the parameters that we know to be more probable than the others. The problem of selecting the prior distribution is not comprehensively addressed here.

If we do not have any a priori knowledge about the parameters, an *uninformative prior* can be used. This is the case in all practical examples and implementations in this thesis. That is, we state $\pi_{pr}(\theta) = 1$. If we have limits for the parameters, we can assign a uniform prior for the parameters in the feasible interval. [6]

For informative priors it is often useful to use so called conjugate priors. This means that both the prior and the posterior come from the same family of distributions. Conjugate priors can be found, for example, for exponential and Gaussian (normal) distributions. The conjugate priors are discussed for example in [3].

2.3.2 Likelihood in Parameter Estimation

As said, in the Bayesian framework the error ϵ in (2.1) is distributed according to some distribution that has some probability density function (PDF), say p_ϵ. If we assume that the measurement error is independent of θ, it can be shown ([2]) that the difference between the measurements and predicted values is distributed in the same way as the error. That is, the likelihood can be written as

$$p(\mathbf{y}|\theta) = p_\epsilon(\mathbf{y} - f(\mathbf{x};\theta)). \tag{2.6}$$

If we assume that the measurement noise is Gaussian with mean zero and covariance C, that is, $\epsilon \sim N(0, C)$, the likelihood can also be written as the Gaussian PDF for the difference between measurements and observations ([2]):

$$p(\mathbf{y}|\theta) = \frac{1}{(2\pi)^{n/2}(\det C)^{1/2}} e^{-0.5(\mathbf{y}-f(\mathbf{x};\theta))^T C^{-1}(\mathbf{y}-f(\mathbf{x};\theta))}. \tag{2.7}$$

Especially, if we assume that the error terms $\epsilon_i = y_i - f(x_i;\theta)$ (measurement error for measurement i) are independent and normally distributed, that is $\epsilon_i \sim N(0, \sigma^2)$ and $\epsilon \sim$

$N(0, \sigma^2 I)$, the likelihood for a certain measurement gets the form

$$p(y_i|\theta) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-0.5\sigma^{-2}(y_i - f(x_i;\theta))^2}. \tag{2.8}$$

Since the error terms are assumed to be independent, the combined likelihood of all the measurements can be written as a product

$$p(\mathbf{y}|\theta) = \prod_{i=1}^{n} p(y_i|\theta) = \frac{1}{(2\pi\sigma^2)^{n/2}} e^{-0.5\sigma^{-2} SS_\theta} \tag{2.9}$$

where $SS_\theta = \sum_i (y_i - f(x_i, \theta))^2$. This is the basis of the practical implementations in this thesis. Note that if measurement errors in different points are not identically distributed or if correlations between error terms exist, the PDF has to be written in full form (equation 2.7).

When using an uninformative prior $\pi_{pr}(\theta) = 1$, also the posterior is known up to the normalizing constant (integral). That is,

$$\pi(\theta|\mathbf{y}) \propto p(\mathbf{y}|\theta). \tag{2.10}$$

2.3.3 Point Estimates

We are often interested, besides in the shape of the posterior distribution, in getting some values that in some sense represent the posterior distribution. We can take the "most probable" value of the posterior density that leads to *maximum a posteriori* value (MAP):

$$\hat{\theta}_{MAP} = \max_{\theta} \pi(\theta|\mathbf{y}). \tag{2.11}$$

For the MAP estimate we normally use the unnormalized posterior $\pi(\theta|\mathbf{y}) \propto p(\mathbf{y}|\theta)\pi_{pr}(\theta)$, since it is simple to evaluate and results to the same estimate. [6], [7]

Now

$$\hat{\theta}_{MAP} = \max_{\theta} p(\mathbf{y}|\theta)p(\theta). \tag{2.12}$$

If we use the non informative prior, the task of finding the MAP reduces to finding the Maximum Likelihood (ML). The estimate is normally abbreviated as MLE (Maximum

Likelihood Estimate) [7]:

$$\hat{\theta}_{ML} = MLE = \max_{\theta} p(\mathbf{y}|\theta). \tag{2.13}$$

In practice maximizing the likelihood is equivalent to maximizing the log-likelihood function $L_{log} = \log(p(\mathbf{y}|\theta))$, which is the same as minimizing $-\log(p(\mathbf{y}|\theta))$. This results in the following target function:

$$-L_{log} = -\log(p(\mathbf{y}|\theta)) = \sum_{i=1}^{n} 0.5\sigma^{-2}(y_i - f(x_i;\theta))^2 + 0.5n\log(2\pi\sigma^2). \tag{2.14}$$

This kind of objective function is often chosen, because it is easier to optimize than the likelihood itself. Also some optimization routines are especially designed for objective functions that contain sums of squares. In addition, the optimization routines often assume that the objective function is to be minimized and that is why the minus sign is used.

2.3.4 Example: Coin Tossing

To illustrate the Bayesian Framework a simple analytical example of coin tossing is presented here (adopted from [4]). Let Y_i represent the result obtained from the i:th toss of a coin so that $Y_i = 0$ means tails and $Y_i = 1$ heads. We are now interested in the probability of getting heads in a series of tosses. Let θ denote the probability of receiving heads. Now we can write the probability of observing a particular series of tosses $\mathbf{y} = (y_1, ..., y_N)$ conditioned on the probability θ:

$$P(y_1, ..., y_N|\theta) = \prod \theta^{y_i}(1-\theta)^{1-y_i} = \theta^{\sum y_i}(1-\theta)^{1-\sum y_i} = \theta^{N_1}(1-\theta)^{N_0} \tag{2.15}$$

where N_1 is the number of heads in the observations and N_0 number of tails. This is the likelihood of receiving a particular series of heads and tails, supposing that the probability for receiving heads is θ. That is, the likelihood contains the information about how well various parameter values $\theta \in [0, 1]$ are able to explain the observed data.

The maximum likelihood estimate is taken as the value that maximizes the log-likelihood function

$$L_{log}(\theta) = \log(\theta^{N_1}(1-\theta)^{N_0}) = N_1 \log(\theta) + N_0 \log(1-\theta). \tag{2.16}$$

The maximum likelihood estimate is $\hat{\theta}_{ML} = \frac{N_1}{N}$, which is the same as the estimate from the Frequentist framework.

If we select a non-informative prior (see section 2.3.1) for different values θ, meaning that we consider all values for the "true" θ in the interval $[0, 1]$ equally possible, we can formulate the posterior density with the Bayes' rule as follows:

$$\pi(\theta|\mathbf{y}) = P(\theta|y_1, ..., y_N) = \frac{\theta^{N_1}(1-\theta)^{N_0}}{\int_0^1 \theta^{N_1}(1-\theta)^{N_0} d\theta} = \frac{(N+1)!}{N_0!N_1!}\theta^{N_1}(1-\theta)^{N_0}. \quad (2.17)$$

Here the integral in the denominator is analytically derived using the beta integral (see [8]). The development of the posterior density function in a series of throws is illustrated in figure 2.1.

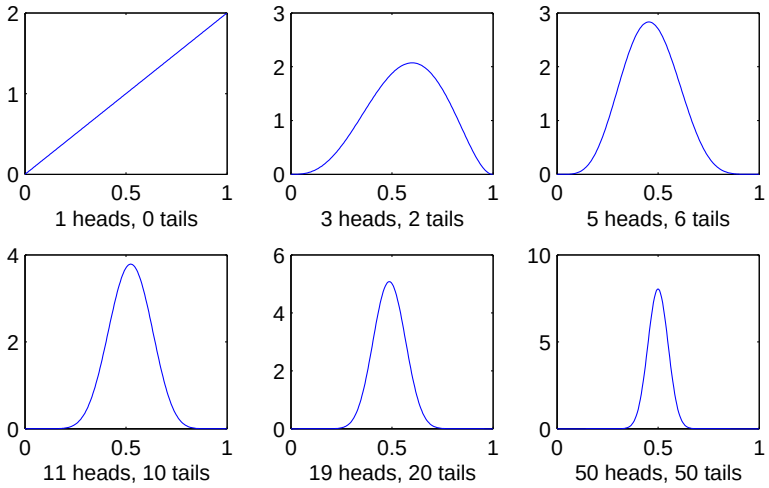

Figure 2.1: The development of the posterior in a set of throws. Note that we get a result even with $N = 1$, when the Frequentist estimate would give either probability 1 or 0.

3 Monte Carlo Methods

The term *Monte Carlo* (MC) method is normally expressed in a very general way - MC methods are stochastic methods; methods that involve sampling random numbers from probability distributions to investigate a certain problem. The Monte Carlo methods are mainly meant for solving two kinds of problems that often arise in statistical analysis. MC methods provide a way to generate samples from a given probability distribution. On the other hand they give a solution to the problem of estimating expectations of functions under some distribution and thus calculating numerical approximations for integrals.

In this section we consider the mathematical background of Monte Carlo methods, explaining why Monte Carlo methods work in the problems stated above and how accurate they are. In addition to the fundamental theory of simulation, we introduce some basic MC methods. The section begins with a short introduction to algorithms for creating random samples from different probability distributions - algorithms that are needed to implement Monte Carlo methods. The theory is based on [6], [9], [10] and [11].

3.1 Sample Generation

A major issue in statistics is the ability to create samples from a given probability distribution. In the Bayesian framework, we want to create samples from the posterior density in order to examine the correlation and accuracy of model parameters and predictions. The Monte Carlo based methods rely on the possibility of creating random variables from arbitrary and possibly complex distributions. In this section the basic methodology of sampling from different distributions, in particular the normal distribution, is first discussed. Then, a different type of numerical sampling method (Accept-Reject), is presented. We assume here that we are able to sample from the uniform distribution $U[0,1]$ and we do not address how this is done in practice.

3.1.1 Traditional Methods

If we know the cumulative distribution function (CDF) of the distribution we want to sample from, we can use the Inverse Transform method (Inverse CDF method) to produce samples from the target distribution. The inverse CDF method is based on the fact that a

random variable $F^{-1}(u)$, where u is sampled from the uniform distribution $U[0,1]$ has the distribution F. That is, random points are "shot" from the y-axis to the CDF curve and the corresponding points in the x-axis are regarded as iid samples from the target distribution (see figure 3.1). [6], [8]

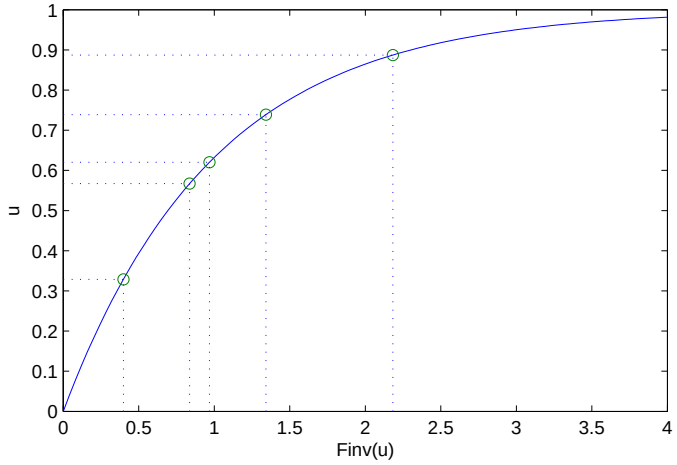

Figure 3.1: Inverse CDF method for producing samples from the exponential distribution with $F = 1 - e^{-x}$ and $F^{-1} = -ln(1-u)$.

The inverse CDF method assumes that the CDF is known. This might be difficult or impossible to calculate analytically, however. In this case an empirical CDF function can be formed by calculating the CDF F in points $\mathbf{x} = (x_1, ...x_n)$ and interpolating a point from interval $[x_{i-1}, x_i]$ that satisfies $x_{i-1} < F(u) < x_i$.

In many applications, including MCMC as formulated in this thesis, it is essential that we can produce samples from Gaussian distributions. Producing random samples $\mathbf{x}$ from a univariate Gaussian $N(\mu, \sigma^2)$ can be simply generated by $\mathbf{x} = \mu + \mathbf{z}\sigma$, where $\mathbf{z} \sim N(0, I)$ (which means that $z_i \sim N(0,1)$). For a multivariate Gaussian $N(\mu, C)$ (C is the covariance matrix) direct sampling means $\mathbf{x} = \mu + R\mathbf{z}$ where $C = R^T R$ and thus the matrix $R = C^{1/2}$ can be formed via the Cholesky decomposition. Here we assume again that we are able to generate samples from $N(0,1)$. In general, the algorithm for creating $x \sim N(\mu, C)$ goes as follows.

- Compute $C^{1/2}$ using the Cholesky decomposition
- Generate $\mathbf{z} \sim N(0, I)$
- Calculate $\mathbf{x} = \mu + C^{1/2}\mathbf{z}$

To show that the method works correctly, we need the equality $Cov(A\mathbf{y}) = A^T Cov(\mathbf{y})A$. This gives

$$\begin{aligned} Cov(\mathbf{x}) &= Cov(C^{1/2}\mathbf{z}) \\ &= (C^{1/2})^T I(C^{1/2}) \\ &= C. \end{aligned}$$

3.1.2 Accept-Reject Methods

For many distributions it is difficult or impossible to do direct sampling with an inverse transform or enough points for reliable empirical CDF are not available. Sometimes the distribution cannot even be presented in a usable form to use the traditional methods introduced in the previous chapter. The selection of methods called Accept-Reject methods only require that we know the analytical form of the target density up to a multiplicative constant. This is the case in sampling from the posterior distribution in the Bayesian framework.

The *fundamental theorem of simulation* ([6]) forms the basis of Accept-Reject sampling methods. Let us consider, for simplicity, a one-dimensional setting where we have a density $f(x)$, which is bounded in the interval $[a, b]$ so that $f(x) < M$ for all $x \in [a, b]$ and which fulfills $\int_a^b f(x)dx = 1$. To create random samples from this density we can "shoot" random points (X, U) to a rectangular area $x \in [a, b]$ and $f(x) \in [0, M]$. The points fulfilling $U < f(X)$ are regarded as samples from a distribution with density $f(x)$ based on theorem 3.1 ([6]).

Theorem 3.1 (Fundamental Theorem of Simulation)
Simulating $X \sim f(x)$ is equivalent to simulating $(X, U) \sim U\{(x, u) : 0 < u < f(x)\}$.

The Accept-Reject method produces samples from a distribution $p(x)$ using an envelope function $q(x)$ that satisfies $p(x) \leq Mq(x)$, where $M < \infty$. Assuming that we can sample from $q(x)$, the Accept-Reject algorithm for producing one sample from distribution with density $p(x)$ goes as follows

1. Generate a candidate point X from proposal function that has unnormalized density $q(x)$ and generate U from $U[0, 1]$.

2. Accept X as a sample of a distribution (with unnormalized density $f(x)$) if $U \leq f(X)/Mq(X)$. If accepted, end the algorithm. Otherwise go to step 1.

The Accept-Reject method, illustrated in figure 3.2, has some limitations. The efficiency of the algorithm depends on how close the proposal distribution is to the distribution from which one wants samples. The constant M often has to be quite large so that the inequality is fulfilled over the whole space, especially with large dimensions. This leads to low acceptance probabilities: $P(x\ accepted) = P(U < \frac{p(x)}{Mq(x)}) = 1/M$. [6], [8], [12]

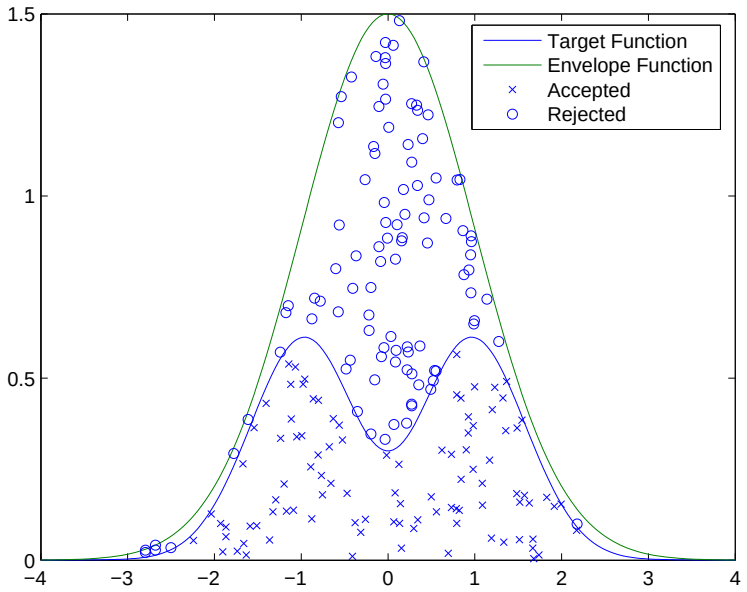

Figure 3.2: Accept-Reject demonstration. Sampling from function $p(x) = e^{-0.5x^2}(\sin^2(x) + 0.3)$ using the envelope function $q(x) = 1.5e^{-0.5x^2}$. 40 points accepted, 60 rejected.

3.1.3 Bootstrapping

In this thesis we are interested in ways to investigate how the distributions of the unknown parameters in a general nonlinear model (equation 2.1) behave. The simplest idea to produce samples from the distribution of the parameters is to add random noise to the data and, at each step, do the LSQ fit and regard the different parameter values as a sample from the posterior distribution. This does not work, however, if the added noise is different from the actual measurement noise. Often we do not know the measurement noise, and each iteration would require a possibly time consuming optimization step, which makes the utilization of the method doubtful. [13]

Bootstrapping is a sample generation method in which we use different combinations of the existing data, with which the estimation is done in an iterative manner. In bootstrapping, a sampling with replacement procedure is carried out: if we have n measurements for design variables X and response variables Y, n new indexes J are randomly chosen from indexes $I = (1, ..., n)$. Then, the original data (X_I,Y_I) is $replaced$ with (X_J,Y_J) and the fitting is done again to get a new sample from the posterior distribution of the parameters. In this work, the bootstrapping method and sampling with replacement is needed in the Population Monte Carlo scheme in chapter 7.

3.2 Monte Carlo Integration

The problem of finding expectations is equivalent to integration, since if we can decompose the integrand, say $h(x)$, into a product of a function $f(x)$ and a probability density function $p(x)$, the definite integral can be written as

$$I = \int h(x)dx = \int f(x)p(x)dx = E_{p(x)}[f(x)]. \tag{3.1}$$

That is, if we can estimate the expectation, we are provided with an estimate for the integral as well. In this section different methods for calculating the integral numerically using a random sampling (Monte Carlo) approach are discussed. A general one-dimensional definite integral $I = \int_a^b h(x)dx = \int_a^b f(x)p(x)dx$ is considered in the examples, for simplicity.

3.2.1 Uniform Sampling - Crude MC

The simplest way to calculate the integral numerically is to use Riemann sums, where the integration interval is divided into n parts of lengths Δx_i ($i = 1...n$). The integral estimate can be calculated with

$$\hat{I} = \sum_{i=1}^{n} h(x_i)\Delta x_i. \tag{3.2}$$

This kind of numerical integration is often done in a deterministic manner by dividing the interval into parts of equal length. That is, $\Delta x_i = \Delta x = (b-a)/n$ for all i. This gives the classical formula $\hat{I} = (b-a)/n \sum_{i=1}^{n} h(x_i)$. The simplest Monte Carlo version of the Riemann sum idea is to take the points in which the division is done randomly

from the uniform distribution: $x_i \sim U[a,b]$. This is sometimes referred to as $crude\ MC$. The approach is, however, often ineffective, because all the parts of the integrand are considered to be equally important with respect to the value of the integral. The Monte Carlo approach in this case also adds some computational complexity, since the function value has to be multiplied with a different Δx in each term of the sum.

3.2.2 Non-uniform Sampling

If the integrand consists of a product of a function and a probability density function (equation 3.1), the power of the Monte Carlo approach is seen more clearly. The Monte Carlo estimate can be derived using random variables $(x_1, ..., x_n)$ drawn from the distribution with density $p(x)$. The Monte Carlo estimate for the integral is now

$$\hat{I} = \frac{1}{n} \sum_{i=1}^{n} f(x_i). \tag{3.3}$$

Using the product notation, we are not limited to sampling from the uniform distribution. If there is a representation $f(x)p(x)$ for $h(x)$ so, that the function $p(x)$ includes the "important" parts of the integrand $h(x)$ (where the value of the integrand is high), the efficiency of the method is improved when compared with the traditional Riemannian approach. The improvement in efficiency is illustrated in the example below.

Example. Let us consider an indefinite integral

$$I = \int_{-\infty}^{\infty} e^{-x^2/2} \left(\sin^2 6x + 3\cos^2 x \sin^2 4x + 1 \right).$$

We notice that the integral is given as a product, where the first term is the (unnormalized) density function of the standard normal distribution with mean 0 and variance 1. The density and the integrand are plotted in figure 3.3.

Now we produce the integral estimate in two different ways. The first one ($\hat{I}_1$) is done by selecting points from a uniform distribution, $x_i \sim U[-4,4]$, since we see that the integrand is close to zero at $|x| = 4$. Then equation (3.2) is used to produce the estimate. The second estimate ($\hat{I}_2$) is produced by taking $x_i \sim N(0,1)$. Since the density is unnormalized, the final result has to be corrected by the inverse of the normalizing constant, $\sqrt{2\pi}$. We can see that the unnormalized density takes into account the important points of the integrand, points close to $x = 0$. The development of the integral estimate with a

different number of sampled points is represented in figure 3.4.

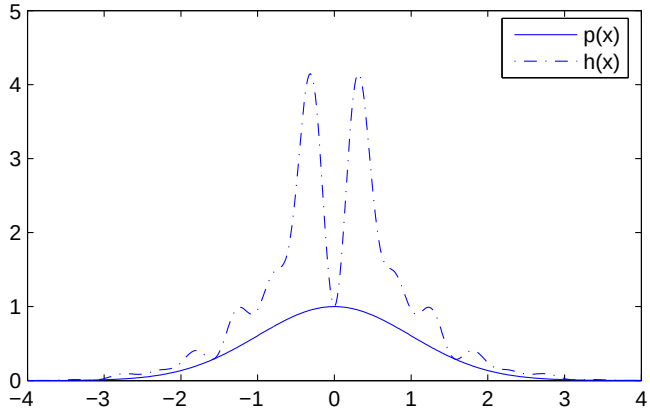

Figure 3.3: Integrand $h(x) = f(x)p(x)$ and (unnormalized) density $p(x)$. The density captures the important parts of the integrand.

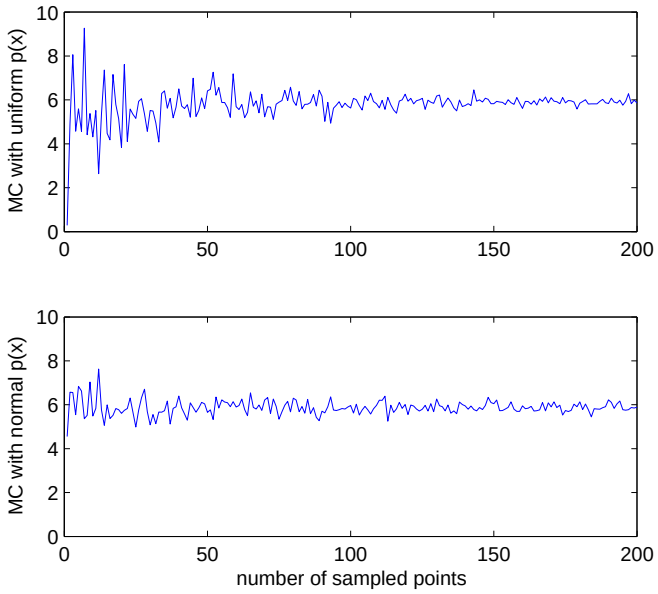

Figure 3.4: Development of estimates $\hat{I}_1$ (crude MC) and $\hat{I}_2$ with respect to increasing sample size. $\hat{I}_2$ seems to converge faster to the correct value.

Note that the integrand $f(x)p(x)$ and the effectiveness of the simple Monte Carlo approach is strongly dependent on the density $p(x)$. Often the integrand is not represented in the product notation, or the distribution with density function $p(x)$ might be difficult to sample from and the distribution might not cover the important parts of the integrand. The integrand can, however, be written in a form of a product of a function and any probability density, that has the desired properties. This idea, called importance sampling, is presented in section 3.2.4.

3.2.3 Hit and Miss Strategies

A simple way to evaluate integrals of a positive function $h(x)$ is to use a similar kind of approach as in Accept-Reject methods for producing random variables from different distributions. Suppose that we have function $Mq(x)$ that satisfies $Mq(x) > p(x)$ for all x and we are able to produce n samples $\{x_i\}$ from the distribution with PDF $q(x)$. Then, for each sampled point we can produce samples y_i from $U[0, Mq(x_i)]$. That is, we have points $\{x_i, y_i\}$ "under" the curve $Mq(x)$. Now we can simply calculate the number of points satisfying $y_i < h(x_i)$. If $Mq(x)$ is chosen so that its integral I_q is easy to calculate, the estimate for the original integral I can be calculated as a ratio $\hat{I} = m/n \cdot I_q$. That is, the Accept-Reject idea can also be used to evaluate integrals.

The simplest way to use this idea to calculate one dimensional integrals of positive functions is to take $q(x)$ to be a constant large enough. Now we can produce n samples $\{x_i, y_i\}$ inside the "box" $a < x < b$, $0 < y < M$ and see which points y_i satisfy $y_i < h(x_i)$ (altogether m points) and simply calculate the integral estimate as a ratio $\hat{I} = m/n \cdot I_q = m/n \cdot (b-a)K$. This is often inefficient, since the area of the box can be large compared to the integral, and many points have to be created to get a reliable estimate.

The "hit and miss" simulation idea can be generalized to many situations - the idea is simply to generate many samples, see which of them satisfy a desired property and calculate ratios. An example of calculating tail probabilities for a normal distribution using this type of simulation is given in section 3.2.4.

3.2.4 Importance Sampling

As seen in section 3.3, the error given by crude Monte Carlo integration converges quite slowly to the true integral value, namely with rate $1/\sqrt{n}$. One of the most popular techniques to reduce the variance of the estimate is importance sampling. The theory is based on [6], [4], [7] and [14].

In crude MC method the random points with which the integral is estimated are often drawn from a uniform distribution. That is, every point in the integral is considered equally important with respect to the value of the integral. In addition, the density $p(x)$ in the integrand (equation 3.1) can be difficult to sample from. In importance sampling one

tries to generate more points from the important regions of the target function that govern the value of the integral, that is, where the integrand has large values.

In importance sampling a density $g(x)$ is introduced. This function roughly estimates the function $h(x) = f(x)p(x)$ in equation (3.1). We can rewrite the integral and the MC estimate as follows

$$\int_a^b f(x)p(x)dx = \int_a^b f(x)\frac{p(x)}{g(x)}g(x)dx \simeq \frac{1}{n}\sum_{i=1}^{n} f(x_i)\frac{p(x_i)}{g(x_i)} = \frac{1}{n}\sum_{i=1}^{n} f(x_i)w(x_i). \tag{3.4}$$

Here the points x_i are drawn from the distribution with $g(x)$ as PDF. That is, we "force" the integrand into a desired product form and by introducing the additional density $g(x)$ we can decide the distribution from which we sample when the integral is estimated. The function $g(x)$ is sometimes referred to as *importance function*, and $w(x)$ as *importance weight*. The $g(x)$ is chosen so that it somehow mimics the target distribution and that it is easy to sample from. The importance function should capture in particular the peaks of the target distribution, and be positive where the target distribution is positive. The analogy to the crude MC method (section 3.2.1) is that with importance sampling we choose the density respect to which the expectation (integral) is calculated. Numerically, the importance function $g(x)$ in equation (3.4) has the same role as $p(x)$ in equation (3.1). Two examples are now given to illustrate the importance sampling approach. The first one compares the importance function to taking points uniformly from the integration interval. The second example illustrates the effectiveness of importance sampling compared to the hit and miss strategy.

Example. To illustrate importance sampling in comparison to crude Monte Carlo integration, let us consider the simple integral $I = \int_0^1 x^{1.6}e^{-x}$. As importance function we choose for example $g(x) = 2.6x^{1.6}$ from which we can sample using the inverse CDF method. The cumulative distribution function for the importance function is $G(x) = x^{2.6}$ and the inverse CDF function $G^{-1}(x) = x^{-2.6}$. Figure 3.5 shows the integrand and the importance function. We see that the importance function weights the points more near the upper bound, where the integrand has its largest values. Figure 3.6 presents the convergence of crude MC and importance sampling with respect to the sample size in this simple example.

Example. Let us consider a task of calculating the probability $P(X > M)$ where $X \sim N(0, 1)$ and M is large so that the probability to be calculated is small.

The basic Monte Carlo approach would be to to sample numbers from $N(0, 1)$ and calcu-

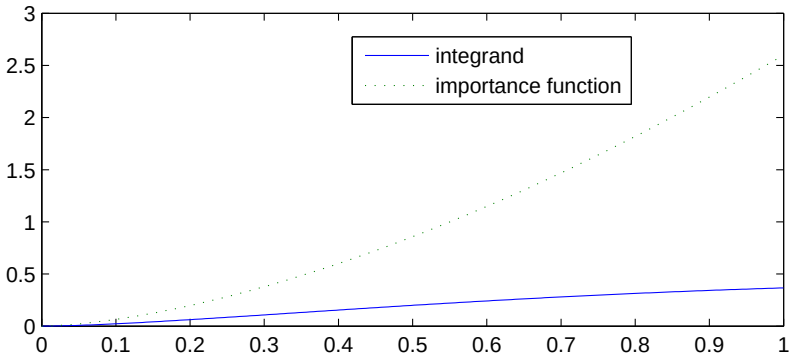

Figure 3.5: Integrand $x^{1.6}e^{-x}$ and importance function $g(x) = 2.6x^{1.6}$.

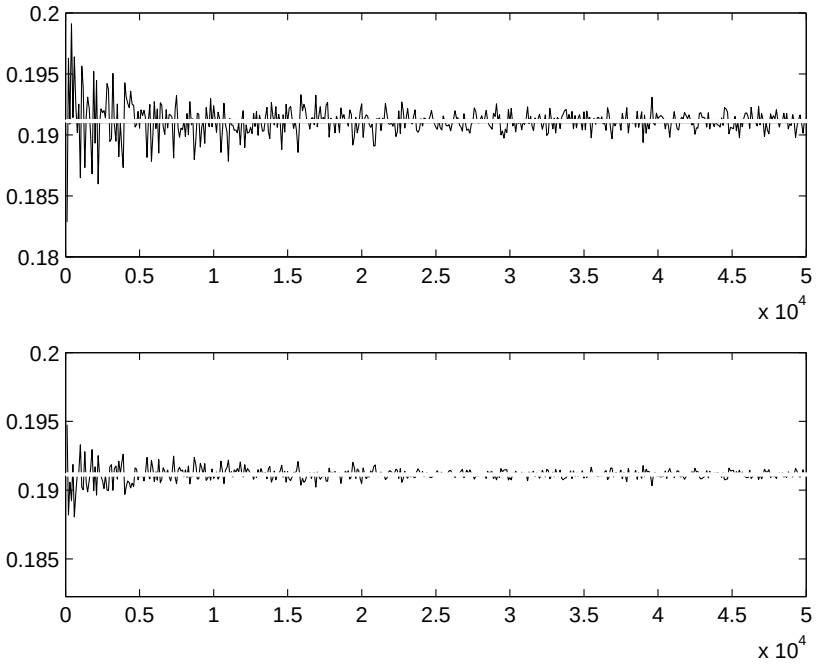

Figure 3.6: Crude MC (up) and importance sampling (down). Importance sampling converges much faster for this example and gives smaller variance.

late the ratio of points satisfying $X > M$. The estimate $\hat{I}_1$ is produced this way. The task is equivalent to calculating the integral of the gaussian PDF from M to ∞, but since we know that the integral from $-\infty$ to ∞ is 1, we can use the simple hit and miss strategy (Monte Carlo simulation).

The estimate $\hat{I}_2$ is produced by importance sampling with importance function $g(x) = e^{-(x-M)}$ where $x > M$, which represents the exponential distribution. That is, we emphasize the important area (the tail of the distribution). The CDF is now $G(x) = 1 - e^{-(x-M)}$ and the inverse of the CDF is $G^{-1}(x) = M - \ln(1-x)$. The inverse CDF method (sec-

tion 3.1.1) is used to produce samples from the importance density. Now we calculate the Monte Carlo estimate for the integral

$$I_2 = \int_M^\infty h(x)dx = \int_M^\infty \frac{h(x)}{g(x)} g(x)dx \approx \sum_{i=1}^{n} \frac{h(x_i)}{g(x_i)}$$

where points x_i are taken from the exponential distribution with PDF given by $g(x)$. The two approaches with $M = 4.5$ are compared in figure 3.7. One can see that the hit and miss approach that is based on ratios does not work in cases, where the ratio is very low compared to the number of samples.

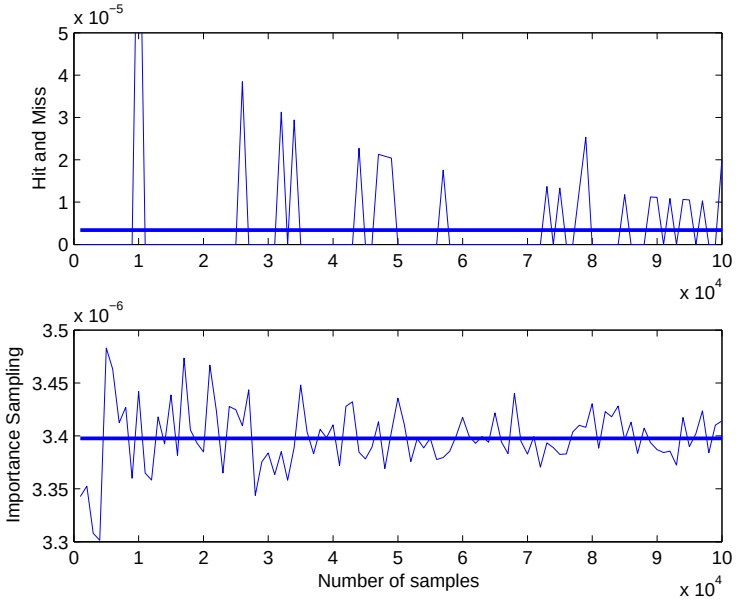

Figure 3.7: Computing the tail probability of a standard normal distribution with hit miss and importance sampling. The blue line represents the correct value.

Many variations and generalizations of the importance sampling idea exist. One group of such methods consists of iterative or sequential importance sampling methods. Another popular method that has been under research recently, is the population Monte Carlo (PMC), which is an iterative approach to importance sampling. PMC is shortly reviewed in chapter 7.

3.3 Convergence of MC estimates

How can we be sure that the Monte Carlo estimate of an integral converges to the right value as the number of samples generated approaches infinity? How fast is the method

converging to the right value and how accurate the estimate approximately is when a certain number of samples are produced? These questions are addressed in this section with two important theorems related to random sampling methods: Laws of Large Numbers and Central Limit Theorem.

3.3.1 Laws of Large Numbers

Roughly speaking, the Laws of Large Numbers (LLN) essentially say, that if we have a sequence of random numbers generated from the same distribution, the average of them gets arbitrary close to the expectation of their common distribution, when the length of the sequence approaches infinity. Normally, in probability theory, two formulations for the laws are given: the Weak Law of Large Numbers (WLLN) and the Strong Law of Large Numbers (SLLN). Here the laws are presented with a short proof for the weak law, and their relevance in justifying MC methodology is explained.

Theorem 3.2 (Weak Law of Large Numbers) *Let $X_1, ..., X_n$ be a sequence of iid random variables with mean μ and finite variance σ^2. Then the sample average*

$$\overline{X}_n = \frac{X_1 + X_2 + ... + X_n}{n}$$

converges in probability to the common mean μ.

Proof. As stated in Appendix 1, the convergence in probability of a random variable $\overline{X}_n$ to μ means that for any number $\epsilon > 0$

$$\lim_{n\to\infty} P(|\overline{X}_n - \mu| < \epsilon) = 1.$$

The Chebyshev inequality for random variables ([8]) states that for a random variable $\overline{X}_n$

$$P(|\overline{X}_n - \mu| \geq \epsilon) \leq \frac{Var(\overline{X}_n)}{\epsilon^2}.$$

Now we have $Var(X_i) = \sigma^2$ for all i and $Cov(X_i, X_j) = 0$ for all (i, j). Thus

$$Var(X_i + X_j) = Var(X_i) + Var(X_j) + 2Cov(X_i, X_j) = Var(X_i) + Var(X_j).$$

Thus, since we know that $Var(aX) = a^2 Var(X)$ ([8]), we get

$$\begin{aligned} Var(\overline{X}_n) &= Var((X_1 + ... + X_n)/n) \\ &= \frac{1}{n^2}\left(Var(X_1) + ... + Var(X_n)\right) \\ &= \sigma^2/n. \end{aligned}$$

Using the Chebyshev inequality we can write

$$P(|\overline{X}_n - \mu| < \epsilon) = 1 - P(|\overline{X}_n - \mu| \geq \epsilon) \geq 1 - \frac{\sigma^2}{n\epsilon^2} \rightarrow 1 \; as \; n \rightarrow \infty$$

which completes the proof.

Theorem 3.3 (Strong Law of Large Numbers) *Let $X_1, ..., X_n$ be a collection of iid random variables with mean μ and $\overline{X}_n$ their sample average. Then*

$$P\left(\lim_{n\rightarrow\infty} \overline{X}_n = \mu\right) = 1. \tag{3.5}$$

That is, the sample average converges $almost\ surely$ to the common mean μ.

The stochastic terms *almost surely*, *convergence in distribution* and *convergence in probability* used in the theorems are explained in more detail in Appendix 1.

We can see that the integral estimate $\hat{I}$ given by equation (3.3) is a sample average of iid samples $(f(x_1), ..., f(x_n))$ that have a common expectation $\mu = E_{p(x)}[f(x)] = I$. Thus, if we consider the Laws of Large Numbers, we can see that $\hat{I} \rightarrow I$ when $n \rightarrow \infty$. That is, the Monte Carlo estimate converges to the correct value of the integral with increasing sample size. The law is the justification of Monte Carlo based simulation methodology (see [6]).

3.3.2 Central Limit Theorem

In estimation tasks it is crucial to know how accurate the estimate is. In basic Monte Carlo estimation we can use the Central Limit Theorem to study the rate of convergence in MC methods. [11], [15]

Theorem 3.4 (Central Limit Theorem) *Let $X_1, ..., X_n$ be a collection of iid random variables with mean μ, finite variance σ^2 and sample mean $\overline{X}_n$. Then*

$$\frac{\overline{X}_n - \mu}{\sigma/\sqrt{n}} \rightarrow N(0,1).$$

The CLT tells that the average of *any* iid random variables converges *in distribution* to the Normal distribution with mean μ and variance σ^2/n. That is, the error of the estimate follows the distribution $N(0, \sigma^2/n)$. This means that the error decreases at rate $n^{-1/2}$. Note that the error does not depend on the dimension of the integral, which justifies the usage of MC methods in high dimensional integrals. The computation time with deterministic numerical integration methods based on discretization increases rapidly when the dimension increases. MC integration can produce estimates of higher dimensional integrals with less computation.

To observe the error in MC methods, it is possible to construct a confidence interval for the estimate $\overline{X}_n$, since we know that with large n the distribution of the error is Gaussian. The confidence interval with risk level α can be written as

$$[\overline{X}_n - \frac{z\sigma}{\sqrt{n}}, \overline{X}_n + \frac{z\sigma}{\sqrt{n}}] \tag{3.6}$$

where z is the point where the cumulative density function of $N(0,1)$ gets value $1 - \alpha/2$. For more on confidence intervals, refer to Appendix 1.

With Monte Carlo methods it is sometimes possible to approximate the normalizing constant in the Bayesian Rule in equation (2.5). MC methods work in more complex and high-dimensional integrals than traditional deterministic methods. When the integrals get very complex the MC methods also run into trouble, however.

The convergence rate of the basic MC method (sometimes referred to as $crude\ MC$) is relatively slow with respect to n - for one additional significant digit of accuracy one needs approximately 100 times more samples. Therefore it is necessary to employ some variance reduction techniques to improve the crude MC method. These include for example stratified sampling, antithetic variates and importance sampling (section 3.2.4).

4 Markov Chain Monte Carlo Methods

In Bayesian analysis for unknown parameters in mathematical models we are often interested in forming the posterior distribution for the parameters. Since this is rarely possible to do analytically, we are satisfied with a number samples from the posterior distribution of the model parameters. To achieve this by applying the Bayes' rule (equation 2.5) one has to integrate over the whole parameter space to calculate the normalizing constant for the posterior density. A numerical approximation can be achieved through Monte Carlo integration methods (see chapter 3). Especially in high-dimensional cases, however, these methods might be problematic. In this chapter Markov Chain Monte Carlo (MCMC) methods are introduced. With MCMC methods, the posterior distribution can be evaluated without having to worry about the problematic normalizing constant of the Bayes' rule.

The chapter begins by introducing the basic theory of Markov Chains needed in analyzing and justifying MCMC methods and their convergence to the right target distribution (posterior). Then, the basic MCMC method, the Metropolis-Hastings algorithm, is discussed.

4.1 Markov Chains

The idea behind the MCMC methods is to create a certain type of Markov Chain that represents the posterior distribution. In this section the basic concepts related to Markov Chains, in the case of a finite state space, and the way they are used in MCMC methods are discussed. For more detailed description related to basics about stochastic processes and Markov Chains, refer to [9], [2] and [4].

A Markov Process is a certain type of discrete time stochastic process. A Markov Chain is a series of states created by a Markov Process. Assume that we have a series of random variables, $(X^{(0)}, X^{(1)}, ...)$. This series is a Markov Chain (produced by a Markov Process), if the value of $X^{(t+1)}$ only depends on the value of the previous state $X^{(t)}$. Formally

$$P(X^{(t+1)} = s_{t+1} | X^{(0)} = s_0, X^{(1)} = s_1, ..., X^{(t)} = s_t) = P(X^{(t+1)} = s_{t+1} | X^{(t)} = s_t) \tag{4.1}$$

where s_i denotes the state of the chain at "time" i.

Let us look at a Markov Process $\{X^{(t)},\ t = 0, 1, 2, ...\}$ that has a finite state space (for simplicity) S (say k possible states) and the states assume values from $\mathbb{R}^d$. That is, $S = \{s_1, s_2, ..., s_k\}$. A state space here means the set of all possible values obtained by the process. We can define an initial distribution (state) as a vector $\lambda(s_i) = P(X_0 = s_i)$.

If the state space is discrete, we can define a transition probability matrix $P = [p_{ij}]$, where p_{ij} denotes the probability of moving from state s_i to state s_j, thus $p_{ij} = P(X^{(t+1)} = s_j | X^{(t)} = s_i)$. The transition probability matrix, also referred to as the Markov Kernel, does not change over time and thus produces a $time\ homogeneous$ Markov Chain. Note that $\sum_j p_{ij} = 1$ for all i. Now, the initial distribution can be updated to produce the next state of the process by multiplying the initial state with the transition probability matrix and summing over all possible initial states λ:

$$P(x_2 = s_j) = \sum_i \lambda(s_i) p_{ij}.$$

The process continues in a similar manner.

We can also define probabilities $p_{ij}^{(n)}$, that stand for moving from state s_i to state s_j using exactly n steps. Formally

$$p_{ij}^{(n)} = P(X^{(m+n)} = s_j | X^{(m)} = s_i). \tag{4.2}$$

The Chapman-Kolmogorov equations define the transition probabilities for an arbitrary number of steps. If we define P_n as the k-step transition matrix, we have

$$P_{n+m} = P_n P_m \tag{4.3}$$

and

$$P_n = P^n. \tag{4.4}$$

Let $\pi_j(t)$ denote the probability that we are at state j and $\pi(t) = \{\pi_j(t), j = 1, ..., k\}$ probabilities for all states at time t. That is, $\pi_j(t) = P(X_t = s_j)$. Then, using equation (4.4) we can write

$$\pi(t) = \lambda P_t = \lambda P^t. \tag{4.5}$$

The Markov Chain has reached its *stationary distribution*, π^*, if applying the transition kernel to current state leads to the same state:

$$\pi^* P = \pi^*. \tag{4.6}$$

Let π_j^* denote the probability for state s_j in the stationary distribution. With this notation, stationarity means $\sum_i \pi_i^* p_{ij} = \pi_j^*$. That is, the stationary distribution is the left eigenvector of the transition probability matrix P, that has the eigenvalue 1.

A Markov Chain is said to be *irreducible*, if, when starting from any starting point x_0, any point is reachable with positive probability with a finite number of steps. That is, the Markov Kernel of an irreducible Markov Process allows free moves all over the state space - all the states *communicate* with each other - the states are "stochastically connected". Formally, $p_{ij}^{(n)} > 0$ for all states for some n.

Another Markov Chain property needed in MCMC theory is the *periodicity* of a chain. A Markov Chain is said to be *periodic* if there are parts of the state space that the process can visit only at regular time intervals. If the chain is not periodic, it is said to be *aperiodic*. Formally, the chain is aperiodic, if $gcd\{n|p_{ij}^{(n)} > 0\} = 1$ (gcd stands for greatest common divisor).

A state s_i in a Markov Chain is said to be *recurrent*, if the probability of returning to s_i is 1. That is, we return to the state surely at some time. In addition, if the expectation of the return time is finite, the state is said to be *positive recurrent*.

Let us assume that a stationary distribution π^* exists. If, regardless of the initial distribution, the distribution of X_n approaches π^* as $n \to \infty$, the π^* is called the *limiting distribution* of the Markov Process. In this case, the Markov Process has a *unique stationary distribution*, which means that the process will end up to the same stationary distribution independent of the initial distribution λ. In this case the chain is called *ergodic*.

The Markov Chain is said to be *reversible* with respect to a distribution π^*, if the so called *detailed balance* condition holds. That is,

$$p_{jk}\pi_j^* = p_{kj}\pi_k^*. \tag{4.7}$$

If a transition kernel that satisfies the detailed balance is found, the process has a stationary distribution. That is, reversibility implies stationarity, since

$$\sum_j p_{jk}\pi_j^* = \sum_j p_{kj}\pi_k^* = \pi_k^* \sum_j p_{kj} = \pi_k^*$$

and thus $\pi^* P = \pi^*$. The detailed balance condition is often used to show that the process with a certain transition kernel results in a stationary distribution. If, in addition, one

can show that the process is irreducible and aperiodic, there exists a unique stationary distribution.

In Markov Chains Monte Carlo methods the idea is to create a Markov Chain using random sampling so that the created chain has the posterior distribution as its unique stationary distribution (limiting distribution). That is, the MCMC methods produce ergodic Markov Chains. In section 4.2 the most basic methods of achieving this, the Metropolis algorithm and its generalization (Metropolis-Hastings algorithm), are introduced. In chapter 4.3 it is shown that the detailed balance condition holds for the Markov process created by the Metropolis algorithm.

Definition A Markov Chain Monte Carlo (MCMC) method for the simulation of a distribution f is any method producing an ergodic Markov Chain whose stationary distribution is f.

The SLLN and CLT theorems (theorem 3.3 and theorem 3.4) also hold with certain assumptions when producing a set of correlated samples using MCMC. That is, a subset of an ergodic Markov Chain can be regarded as a set of iid random variables, when the MCMC algorithm has been run long enough. That is, the MCMC estimate of the average of the chain converges to the "true" mean of the distribution - the sampled values asymptotically approach their correct values:

$$\hat{f}_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i) \approx \int_{\mathbb{R}^d} f(x)\pi(dx). \tag{4.8}$$

The convergence theorems can be written from the perspective of Markov Chains as follows ([15]).

Theorem 4.1 (SLLN for Markov Chains) *Suppose $\{X_j\}_{j=1}^{\infty}$ is ergodic with stationary distribution π. Then, if f is real and $\pi|f|$ bounded, $\hat{f}_n \to f\pi$ with probability 1.*

Theorem 4.2 (CLT for Markov Chains) *Suppose that the assumptions in theorem 4.1 hold. Then there exists a real number $\sigma^2(f)$ so that*

$$\sqrt{n}(\hat{f}_n - f\pi) \to N(0, \sigma^2(f))$$

in distribution, independent of the initial state.

That is, if the transition kernel in the MCMC method is defined so that it produces an ergodic Markov Chain, the most important convergence laws hold and the produced chain, when run long enough, can be regarded as a set of samples from the target distribution in spite of the Markovian nature of the method. The CLT implies that the sample path averages of the target function f converges towards a Gaussian distribution, and thus provides a measure of the variability of the created states when n is large. For proofs, generalizations and more detailed examination on Markov Chain theory behind MCMC, also in general state space, refer to [2] and [1], for example.

4.2 Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm prescribes a simple transition kernel to produce a markov chain that has invariant distribution $\pi(\theta)$ that can be regarded as a sample from $\pi(\theta)$. The MH algorithm is based on accept-reject methodology: a new candidate point θ^* is created from a proposal distribution $q(.|\theta)$ that contains the probabilities for receiving a certain candidate point given the previous value θ. The Metropolis-Hastings algorithm ca be written as follows:

1. Initialization
   - Choose a starting point θ_0
   - Set $\theta_{old} = \theta_0$
   - Set $Chain(1) = \theta_0$ and $i = 2$
2. Choose a new candidate from the proposal distribution: $\theta^* \sim q(.|\theta_{old})$
3. Accept the candidate with probability

$$\alpha = \min\,(1, \frac{\pi(\theta^*)q(\theta_{old}|\theta^*)}{\pi(\theta_{old})q(\theta^*|\theta_{old})}) \tag{4.9}$$

   - If accepted set $Chain(i) = \theta^*$ and $\theta_{old} = \theta^*$
   - If rejected set $Chain(i) = \theta_{old}$
4. Set $i = i + 1$ and go to 2

If we assume a symmetric proposal distribution, that is, $q(\theta^*|\theta_{old}) = q(\theta_{old}|\theta^*)$, we get a special case of the MH algorithm called the Metropolis algorithm that was introduced

earlier (1953) by Nicholas Metropolis in [16]. For additional representations of the algorithm, refer to [6], [1], [9] or [3].

Note here that if we use the Bayesian framework we know the posterior density $\pi(\theta)$ up to a normalizing constant. We see that in equation (4.9) the constant cancels out. If we assume a standard nonlinear model ($\mathbf{y} = f(\mathbf{x};\theta)+\epsilon$) with Gaussian noise ($\epsilon \sim N(0,\sigma^2 I)$) and a non-informative prior $\pi_{pr}(\theta) = 1$, we can write the acceptance probability for the Metropolis algorithm as follows:

$$\alpha = \min\left(1, \frac{\pi(\theta^*)}{\pi(\theta_{old})}\right) = \min\left(1, \frac{p(y|\theta^*)}{p(y|\theta_{old})}\right) = \min\left(1, e^{-0.5\sigma^{-2}(SS_{\theta^*} - SS_{\theta_{old}})}\right). \quad (4.10)$$

This is a practical form of the acceptance rule and it is the basis of the implementations of different MCMC methods in this work. The third step (acceptance step) of the algorithm can now be written in a more practical way as follows

3. Compute SS_{θ^*} and $SS_{\theta_{old}}$. Accept the candidate if
$SS_{\theta^*} < SS_{\theta_{old}}$ or if $u < e^{-0.5\sigma^{-2}(SS_{\theta^*} - SS_{\theta_{old}})}$
where u is a random number generated from $U[0,1]$.

Many modifications of the basic MCMC based on the Metropolis-Hastings algorithm have been developed. The promising family of adaptive methods is discussed in chapter 6. In addition, several aspects have to be considered when using MCMC in practice, as well as when evaluating the convergence of the algorithms. These issues are addressed in chapter 5.

4.3 Metropolis-Hastings as a Markov Chain

Why does the Metropolis-Hastings algorithm work - how can we show that it produces a Markov Chain with a stationary distribution $\pi(\theta)$? Here the detailed balance condition (equation 4.7) is considered, which is a sufficient (not necessary) condition for the existence of a unique stationary distribution with a desired probability density function. The detailed balance condition can be shown to hold for the transition kernel produced by the MH algorithm as presented below. In [2] it is shown in detail, that the metropolis kernel with some simple proposal distributions, such as Gaussian with fixed covariance, centered at the previous point, produces a Markov Chain that has the necessary ergodicity properties.

The goal is to form the transition kernel $P = [p_{ij}]$ for the MH algorithm that describes the probabilities of moving from state s_i to state s_j. Then we need to show that for the transition kernel the DB condition holds, that is $\pi(s_i)p_{ij} = \pi(s_j)p_{ji}$. The transition kernel can be formed in the following way:

$$\begin{aligned} p_{ij} &= P(moving\ from\ s_i\ to\ s_j) = P(proposing\ s_j)P(accepting\ s_j) \\ &= q(s_j|s_i)\alpha(s_i, s_j) = q(s_j|s_i)\min\left(1, \frac{\pi(s_j)q(s_i|s_j)}{\pi(s_i)q(s_j|s_i)}\right). \end{aligned}$$

Thus,

$$\begin{aligned} \pi(s_i)p_{ij} &= \pi(s_i)q(s_j|s_i)\min\left(1, \frac{\pi(s_j)q(s_i|s_j)}{\pi(s_i)q(s_j|s_i)}\right) \\ &= \min\left(\pi(s_i)q(s_j|s_i), \pi(s_j)q(s_i|s_j)\right) \\ &= \pi(s_j)q(s_i|s_j)\min\left(\frac{\pi(s_i)q(s_j|s_i)}{\pi(s_j)q(s_i|s_j)}, 1\right) \\ &= \pi(s_j)p_{ji}. \end{aligned}$$

That is, based on the detailed balance condition, the Metropolis Hastings algorithm produces a transition rule with which the chain has a stationary distribution.

In the case of a discrete state space, the transition probability kernel produced by the Metropolis-Hastings algorithm can be defined as

$$\begin{aligned} p_{ij} &= q(s_j|s_i)\min\left(1, \frac{\pi(s_j)q(s_i|s_j)}{\pi(s_i)q(s_j|s_i)}\right), \quad (i \neq j) \\ p_{ii} &= 1 - \sum_{j \neq i} p_{ij} \end{aligned}$$

which satisfies the detailed balance condition.

Another way to justify that the Metropolis algorithm leads to a certain stationary distribution is to examine the situation, where we have two arbitrary members of the state space, say s_a and s_b, so that $\pi(s_b) \geq \pi(s_a)$. Let us assume that at time $t-1$, we have a draw from the posterior: $s_{t-1} \sim \pi$. Now we can show that we can always make an exact move with the Metropolis kernel between two arbitrary states and the probability of movement is the same in both directions.

First, let us consider the situation, where we move from s_a to s_b. That is, we examine the

probability that we are at state s_a at time $t-1$ and at state s_b at time t:

$$\begin{aligned} P(s_t = s_b; s_{t-1} = s_a) &= P(s_{t-1} = s_a)P(s_t = s_b | s_{t-1} = s_a) \\ &= \pi(s_a)q(s_b|s_a)\alpha(s_a, s_b) \\ &= \pi(s_a)q(s_b|s_a) \end{aligned}$$

because for a move upwards the point is always accepted ($\alpha = 1$).

For the move $s_b \rightarrow s_a$, we get

$$\begin{aligned} P(s_t = s_a; s_{t-1} = s_b) &= \pi(s_b)q(s_a|s_b)\alpha(s_b, s_a) \\ &= \pi(s_b)q(s_a|s_b)\frac{\pi(s_a)}{\pi(s_b)} \\ &= \pi(s_a)q(s_b|s_a) \end{aligned}$$

for a symmetric proposal $q(x|y) = q(y|x)$, as is the case in the Metropolis algorithm.

That is, we see that $P(s_t = s_b; s_{t-1} = s_a) = P(s_t = s_a; s_{t-1} = s_b)$ for any s_a and s_b belonging to the state space. This implies that $\pi(s_{t-1}) = \pi(s_t)$ and that π is the stationary distribution of the process. A similar calculation can be carried through for the more general Metropolis-Hastings kernel, where $q(x|y) \neq q(y|x)$ is allowed. Essentially, the idea is based on stationarity caused by reversibility, that is characterized with the detailed balance equation.

The detailed balance condition presented above only quarantees that the created stochastic process has a stationary distribution. For exact investigations about whether the MCMC methods have the ability to converge towards the desired stationary distribution, the terms *geometric ergodicity* and *uniform ergodicity* are often used. It can be shown, using the definition of the so called *coefficient of ergodicity*, that if the created chain is uniformly ergodic, the strong law of large numbers holds and the method converges to the stationary distribution. In addition, it can be shown that the Metropolis-Hastings algorithm with a Gaussian proposal distribution (used in this work) creates a uniformly ergodic Markov chain. See [2] and [17] for proofs and convergence calculations.

4.4 Single Component Metropolis-Hastings

In the Metropolis-Hastings algorithm presented in section 4.2, all components of the chain (values for unknown parameters) are updated at the same time. In practice, when using a Gaussian proposal, this means sampling from a multivariate Gaussian distribution. In the *Single Component Metropolis Hastings* algorithm (SC), presented in the original paper by Metropolis et.al. ([16]), the chain is updated component by component. Thus, θ_t^i represents the t:th sampled value for component i.

That is, the parameter vector is divided into components and one iteration of the algorithm contains p steps where p is the length of the parameter vector. This leads to p different conditional proposal distributions and p different acceptance probabilities. The proposal distribution for component i at iteration t can be univariate Normal distribution with center at the previously sampled point θ_{t-1}^i and some fixed variance σ_i^2. That is, $q_t^i \sim N(\theta_{t-1}^i, \sigma_i^2)$. In the acceptance probability calculation the posteriors are used. The difference to MH is that when sampling component i, all previously sampled components $(1, ..., i-1)$ from the same iteration are used in evaluating the posterior. That is, at iteration t, the posterior is made one-dimensional by fixing components $(1, ..., i-1)$ with values obtained from iteration t and components $i+1, ..., n$ with values obtained from iteration $t-1$.

This makes SC appealing in high-dimensional cases, because the proposal distributions stay simple and they are easy to sample from. The SC algorithm is shortly presented below (replace steps 2-3 in the Metropolis-Hastings algorithm in section 4.2). See [9] and [2] for more on the topic.

For i = 1 to p (number of parameters)

- Sample $\theta_i^* \sim q_t^i$ (where t represents the index of the sample)
- Accept with probability

$$\alpha = \min\left(1, \frac{\pi(\theta_t^1, ..., \theta_t^{i-1}, \theta_i^*, \theta_{t-1}^{i+1}, ..., \theta_{t-1}^n)}{\pi(\theta_t^1, ..., \theta_t^{i-1}, \theta_{t-1}^i, \theta_{t-1}^{i+1}, ..., \theta_{t-1}^n)}\right)$$

  - If accepted set $\theta_t^i = \theta_i^*$
  - Else set $\theta_t^i = \theta_{i-1}^i$

All methods using one-dimensional proposals and component by component updating of the chain, including SC, Gibbs sampling (section 4.5) and SCAM (section 6.2), may suffer from poor mixing if the parameters are correlated. This is natural, since the correlations are not taken into account in one-dimensional proposal distributions, whereas the covariance matrix in multivariate Gaussian contains this information. That is, the path in pairwise scatter plots produced with SC contains only paths parallel to the coordinate axes - the algorithm can move only along a coordinate axis.

One way to go around the problem of unknown correlations is to rotate the proposal distribution at some predefined steps. This can be done by calculating the covariance matrix of the chain created so far and computing the principal components of the covariance matrix. These directions can be used as sampling directions in the SCAM algorithm. See appendix 1 for more details. [18]

4.5 Gibbs Sampling

Gibbs sampling is a modification of the SC algorithm and it is widely used in different applications. In the Gibbs algorithm the sampling is also done component by component using one dimensional full conditional posterior distributions $\pi(\theta_i|\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_p)$. Gibbs sampling assumes that these conditional distributions are known, which means that all other components than the one to be sampled are fixed. In Gibbs sampling the created points are always accepted. Gibbs algorithm can be used when the conditional distributions can be found easily and are easy to sample from. The algorithm is presented below. [3]

For j = 1 to p (number of parameters)

- Sample $\theta_j^i \sim \pi(\theta_i|\theta_1, ..., \theta_{i-1}, \theta_{i+1}, ..., \theta_p)$
- Set $Chain_i(j) = \theta_j^i$

The construction of the one-dimensional conditional distributions might, however, be complicated. If the conditional distributions are not known in analytical form, an empirical distribution can be created by evaluating the target distribution $\pi(\theta)$ with respect to a given coordinate a number of times. After that, the inverse CDF method (section 3.1.1) can be applied. This requires several iterations, however. [13]

5 On Implementing MCMC

Several issues related to implementing the MCMC methods in practice are discussed here. First, the question of choosing the proposal and tuning its parameters to form an effective proposal distribution is taken into consideration. We try to find ways to assure that the sampling really covers as much of the parameter space as possible.

Secondly, the convergence problem related to MCMC results is introduced: how can we be sure that the method has actually converged so that we can rely that the produced chain is a representative sample of the posterior distribution. This problem is the target of a large amount of current research and the goal here is to present some of the ideas for convergence diagnostics.

Finally, some other practical issues related to implementing MCMC are discussed. These include using a burn-in period, thinning the chain and the possibility of running multiple parallel chains.

5.1 Proposal Distribution

The MH algorithm itself is very simple. However, the performance of the algorithm is dependent on how we choose the proposal distribution $q(.|\theta_{old})$. The proposal distribution has a large effect on the *mixing* of the chain, which means how well the samples are spread over the parameter space and its important parts. It is clear that with too wide a proposal distribution many of the candidate points are rejected and the chain "stays still" for long periods and the target distribution is reached slowly. Then again, when the proposal distribution is too narrow, the acceptance ratio is high but a representative sample of the target distribution is achieved slowly. The term mixing and the effect of the width of the proposal are illustrated in figure 5.1. [2]

There are two basic ways of constructing the proposal distribution. In the first approach we choose a fixed proposal $q(.|\theta_{old}) = q(\theta)$ that is independent of the previous state (parameter values). This is called Independent Metropolis-Hastings and it is very similar to the Accept-Reject algorithm presented in chapter 3.1.2.

A more practical approach takes the previously simulated value into account when the proposal is formed. That is, we perform a local search for new candidate points at the

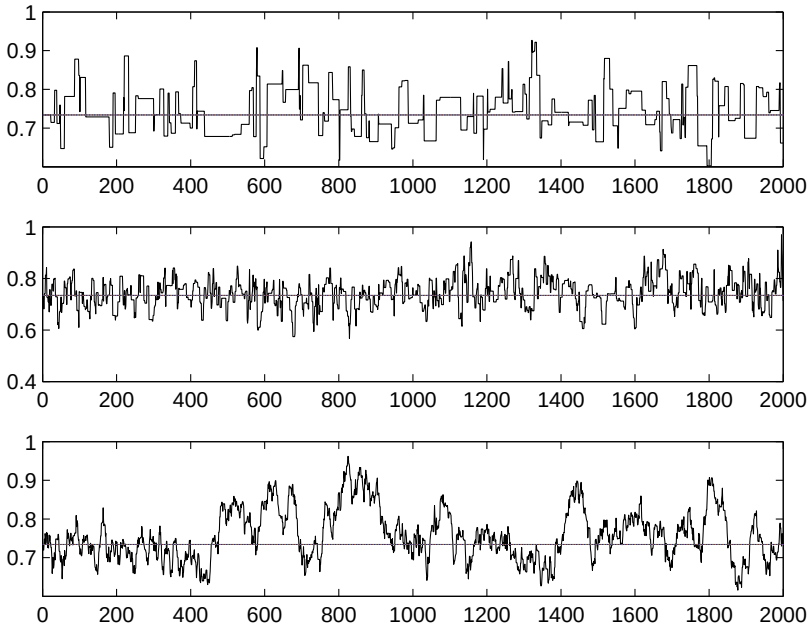

Figure 5.1: Example of a chain of one parameter started with LSQ-optimized parameter values. The upper picture tells that the proposal is too wide - the chain stays still for long periods. The lowest picture presents narrow proposal - the samples don't seem to converge well. The chain in the middle is well mixed. For this chain the MSE was used to calculate the covariance of the proposal.

neighborhood of the current value. This is why this algorithm, called the Random Walk Metropolis algorithm, is in some sources referred to as Local Metropolis, whereas the previous approach sometimes goes with the name Global Metropolis. The Random Walk approach is less dependent on the form of the proposal distribution in relation to the form of the target distribution. It is also more practical since we can use a standard, easy-to-sample proposal distribution, for example a Gaussian distribution centered at the current point. Here we regard the Random Walk Metropolis-Hastings as the basic MCMC algorithm used in different case examples in chapter 9. The Random Walk Metropolis-Hastings algorithm with a Gaussian proposal distribution is presented below.

1. Initialization

   - Choose θ_0, set $\theta_{old} = \theta_0$
   - Choose covariance C
   - Choose N (chain length) and set $i = 1$.

2. Acceptance Step (Metropolis Step)

- Sample θ_{new} from $N(\theta_{old}, C)$ and u from $U[0,1]$
- Calculate SS_{old} and SS_{new}
- If $SS_{\theta_{new}} < SS_{\theta_{old}}$ or $u < e^{-0.5\sigma^{-2}(SS_{\theta_{new}} - SS_{\theta_{old}})}$ set $\theta_i = \theta_{new}$. Otherwise set $\theta_i = \theta_{old}$

3. If $i < M$ set $i = i + 1$ and go to step 1. Otherwise stop the algorithm.

In the algorithm the proposal "width" parameter is the covariance C of the Gaussian proposal distribution, or variance in a one-dimensional case. The problem in choosing the proposal distribution then turns into the problem of tuning the covariance matrix so that the sampling is efficient. Traditionally this is done by choosing a fixed covariance by hand by the modeler using some approximation or "trial and error" strategy. Recently some new data-driven modifications to the basic Metropolis algorithm have been introduced in order to update the covariance matrix as the algorithm proceeds (chapter 6).

Note that the created Markov Process has a stationary distribution, since the detailed balance condition holds. In addition, the stationary distribution is unique, since the random walk process clearly is irreducible - the probability of moving from any point to any other point is always positive. Since the process is random, it lacks all periodic behavior and is thus also aperiodic.

5.2 Initializing MCMC

If we use the Random Walk Metropolis algorithm with a Gaussian proposal distribution, we have to come up with a guess for the Covariance Matrix C. Also for the convergence rate it is useful to choose the starting point θ_0 correctly.

In the MCMC implementations in this work the starting point is chosen to be the point that suits the data in an optimal way in LSQ sense. That is, with a general nonlinear model (i denotes the measurement index)

$$\theta_0 = \min_{\theta} \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2.$$

This leads to a nonlinear optimization task (to a nonlinear LSQ problem to be precise). These can be solved with many methods introduced for example in [19]. In unconstrained nonlinear optimization the polytope search method is one of the most popular, since one

does not need gradients or Hessians. Specifically for nonlinear LSQ, the Levenberg-Marquardt method is probably the most popular. The optimization task can be handled with standard optimization routines found in different scientific computing environments.

The covariance of the Gaussian proposal can be chosen by trial and error. Often, however, it is useful to use the covariance approximation obtained from linearization. That is, the model is linearized and then the formula from linear theory, $\hat{C} = \sigma^2(X^T X)^{-}1$ (see appendix 1 for proof), is used. The linearization is based on Taylor expansion around the estimated point :

$$l(\theta) = l(\hat{\theta}) + \nabla l(\hat{\theta})(\theta - \hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^T H(\theta - \hat{\theta}) + ... \tag{5.1}$$

where H is the Hessian matrix containing the second order derivatives (see Appendix 1). When we consider the LSQ function

$$l(\theta) = \sum_{i=1}^{n} (f(x_i, \theta) - y_i)^2$$

we get expressions for the derivatives at $\theta = \hat{\theta}$:

$$\frac{\partial l(\hat{\theta})}{\partial \theta_i} = 2 \sum_{k=1}^{n} \frac{\partial f(x_k, \hat{\theta})}{\partial \theta_i} (f(x_k, \hat{\theta}) - y_k),$$

$$\frac{\partial^2 l(\hat{\theta})}{\partial \theta_i \partial \theta_j} = 2 \sum_{k=1}^{n} \left(\frac{\partial^2 f(x_k, \hat{\theta})}{\partial \theta_i \partial \theta_j} (f(x_k, \hat{\theta}) - y_i) + \frac{\partial f(x_k, \hat{\theta})}{\partial \theta_i} \frac{\partial f(x_k, \hat{\theta})}{\partial \theta_j} \right).$$

Since the residuals ($f(x_k, \hat{\theta}) - y_k$) are relatively small (the LSQ has been optimized with an optimization routine to produce the estimate $\hat{\theta}$), we can omit them and write an approximation for the Hessian matrix as follows:

$$H_{ij} \approx 2 \sum_{k=1}^{n} \frac{\partial f(x_k, \hat{\theta})}{\partial \theta_i} \frac{\partial f(x_k, \hat{\theta})}{\partial \theta_i}.$$

In matrix form, $H \approx 2J^T J$, where J is the Jacobian matrix calculated at $\hat{\theta}$, containing the first order derivatives (see Appendix 1).

When we insert the approximation to the truncated Taylor expansion (first three terms), we get

$$l(\theta) \approx l(\hat{\theta}) + (\theta - \hat{\theta})^T J^T J (\theta - \hat{\theta}) \tag{5.2}$$

because $\nabla l(\hat{\theta}) \approx 0$ at the LSQ optimum.

In linear case (see appendix 1), when X denotes the design matrix, we have

$$l(\theta) = (X\theta - \mathbf{y})^T(X\theta - \mathbf{y}) = \theta^T X^T X\theta - \theta^T X^T \mathbf{y} - \mathbf{y}^T X\theta + \mathbf{y}^T\mathbf{y} \tag{5.3}$$

which can be written as a quadratic form containing $(\theta - \hat{\theta})$:

$$l(\theta) = (\theta - \hat{\theta})^T X^T X(\theta - \hat{\theta}) + D \tag{5.4}$$

where D is independent of θ. Substituting $\theta = \hat{\theta}$ yields $D = l(\hat{\theta})$ and

$$l(\theta) = l(\hat{\theta}) + (\theta - \hat{\theta})^T X^T X(\theta - \hat{\theta}) \tag{5.5}$$

where is the estimate from the linear theory (see Appendix 1).

Thus, comparing equations (5.2) and (5.5), we see that the Jacobian matrix J assumes the role of the design matrix X in the linear formulas, and the covariance approximation, when the assumption $Cov(y) = \sigma^2 I$ is made, gives

$$C = Cov(\hat{\theta}) \approx \sigma^2 (J^T J)^{-1}.$$

Here the variance σ^2 could be calculated from replicated measurements. This is not often possible, however. We can use the residuals of the fit to estimate the variance, based on the assumption that measurement error is about equal to residuals. [13]

$$\sigma^2 \approx \sigma^2_{MSE} = \frac{RSS}{n-p} = \frac{\sum_{i=1}^{n} (y_i - f(x_i, \theta))^2}{n-p} \tag{5.6}$$

The Jacobian J might be difficult to calculate analytically so we often need a numerical approximation for it, calculated using finite differences.

Finally, the basic Random Walk Metropolis algorithm can be given in pseudo code, from which it is straightforward to implement with a number of programming techniques.

1. (Initialization)

   - Choose N_{simu}
   - Set $\theta_1 = \min_\theta \sum_{i=1}^{n} (y_i - f(x_i, \theta))^2$ (Use some optimization routine)

- Calculate $MSE = RSS/(n-p)$ where n is the number of measurements and p the length of θ
- Set $SS_{old} = SS_{\theta_1}$
- Calculate Jacobian J (numerically or analytically)
- Calculate $C = (J^T J)^{-1} * MSE$
- Calculate R so that $C = R^T R$ (Cholesky decomposition)

2. (Simulation Loop)

- For $i = 2$ to N_{simu}
  - Sample $z = \{z_i\}$, $i = 1...p$ where $z_i \sim N(0, 1)$
  - Set $\theta_{new} = \theta_{old} + Rz$
  - Sample u_α from $U[0, 1]$
  - Calculate SS_{new}
  - Calculate $\alpha = \min(1, e^{-0.5\sigma^{-2}(SS_{new} - SS_{old})})$
  - If $u_\alpha < \alpha$
    - Set $\theta_i = \theta_{new}$
    - Set $\theta_{old} = \theta_{new}$
    - $SS_{old} = SS_{new}$
  - Else
    - Set $\theta_i = \theta_{old}$
  - Endif
- Endfor

Note that the model might have more than one response components that are observed (every measurement y_i is a vector). In this case the calculation of the sum of squares also leads into a vector. The final SS -value can be calculated as the sum of the components.

5.3 Burn-In and Practicalities

In the beginning of MCMC sampling there may be a period before the algorithm converges to the correct distribution and starts producing samples from it. The length of the initial period depends on the shape of the target distribution and the initial values θ_0. The

period in which the chain has not yet converged must be discarded in order to avoid unrepresentative, "false" samples. This period is called the burn-in period. In practice one might give the length of the burn-in period as input to the algorithm, for example 1000 iterations. During the burn-in period the parameter values are not saved into the chain structure. [2]

How long should the burn-in period be? Some ways to detect convergence and end the burn-in period are introduced in section 5.4. The most simple way is to run the MCMC algorithm a few times with different starting values and visually see where the parameter values converge to some equilibrium and thus base the length of the burn-in period on these observations.

One typical option to form the chain is to run a single long run, discard the burn-in period and regard the rest as a sample from the target distribution. Here, however, the samples are not independent, since the sampling is based on a Markov Process and correlation between consecutive members in the chain exists. To reduce the correlation of the samples it is possible to perform thinning, which means saving only every n:th point of the chain. The level of independence can be studied in many ways, of which some are explained in section 5.4.

Other strategies for MCMC sampling include making several medium-length parallel MCMC runs. Here, however, we might run into convergence problems, if the algorithm does not converge fast enough. If we make every parallel chain long, we end up with issues of computational complexity. The task of running parallel MCMC runs would therefore be an interesting application of parallel computing - the parallelization of the problem is quite straightforward. That is, if we have an effective parallel computing environment (a cluster for example), we might be able to run parallel chains with no significant increase in computational time.

The extreme strategy would be to make a large number of short MCMC runs from different (random) initial values and record only the final state of every run. In this work, however, the strategy of one, sufficiently long run is used.

5.4 Convergence Diagnostics and Chain Length

A difficult question in MCMC methods is whether the created chain is long enough so that it has reached its invariant distribution. Another issue that needs consideration is whether

the algorithm has covered the target distribution sufficiently.

How can the sufficient chain length be assessed in MCMC methods? The methods that address this issue belong to the field of MCMC convergence diagnostics. Convergence diagnostics here means statistical analysis done in order to asses convergence of the MCMC algorithm. A nice review about the methods currently available in MCMC convergence diagnostics can be found from [20]. Some of the methods are briefly discussed here.

The convergence assessment issue is difficult in MCMC algorithms, because the rate of convergence vary depending on the algorithm used and the target distribution. It is stated in [20] that it is not possible to construct effective analytical estimates for the convergence rate and accuracy of MCMC algorithms. That is, we cannot get any analytical formula or stopping criteria for the algorithm, that would uniquely determine the run length. The MCMC algorithms can be falsified, but not verified - we can never be totally sure that the sample created is a comprehensive representation of the posterior distribution. Convergence diagnostics are methods for making educated guesses about the convergence of the algorithms.

The methods for assessing convergence can be roughly divided into categories. The most simple and straightforward methods seem to be based on monitoring the created chain visually or with some statistical tools. These methods are quite easy to implement and do not cause a lot of extra computational complexity. That is why these methods are discussed here in more detail than other, more formal methods.

The most obvious method for assessing convergence in MCMC is the visual study of the marginal paths. That is, one can plot every column of the chain separately for every parameter versus the index of the row in question. From marginal chain paths one can see where the values start to converge to a certain level. The largest initial period of the marginal paths, where the chain seems not to have converged, can be regarded as the minimum length for the burn-in period. This is illustrated (for an example task presented in section sec:boxo) in figure 5.2. [2]

The sum of squares values given by every sampled parameter vector can also be studied to get information about the movement of the algorithm. The range in which the SS-values vary should stay about constant when the algorithm has converged.

A popular approach to diagnose convergence is to run a number of parallel chains with very dispersed starting values. Again one can visually study the period, after which the

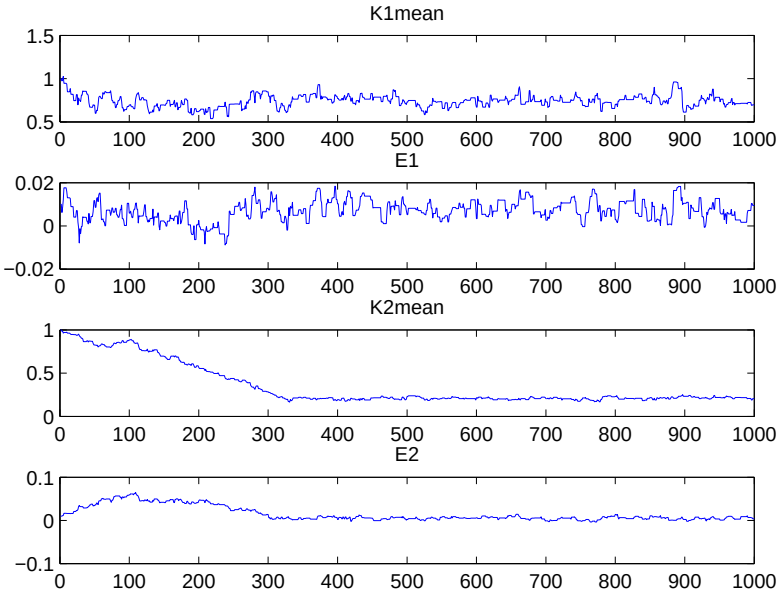

Figure 5.2: Marginal paths for a kinetic model (see section 9.2 for model description). It seems that the first two parameters converge quite fast to their marginal stationary distributions, whereas for the last two the convergence takes 300-400 steps. Thus, the burn-in period should be at least 400 here.

chains start to agree. For visual study one can use the marginal paths, for example. It is also possible to compare the parallel chains using some similarity measure, as explained in [20].

The most simple statistics normally calculated from the MCMC steps is the acceptance ratio, that is the percentage of points accepted until a certain number of steps. Some ranges for "optimal" acceptance ratio are suggested in literature, but since the optimal acceptance ratio is dependent on the shape of the posterior distribution, it is difficult to construct any generally applicable rules concerning the acceptance ratio. However, a value between 0.1 and 0.5 is normally regarded to be satisfactory. Monitoring the acceptance ratio and tuning the proposal distribution according to it is one of the simplest ways to improve the convergence of the MCMC methods. If the acceptance ratio seems too low, a smaller proposal distribution can be tried, for example by scaling the covariance of the Gaussian proposal down, and vice versa for too high acceptance ratios.

The correlation between states that are close to each other causes bias in the estimates based on the produced chain. That is, when assessing if the chain really is a set of iid samples from the target distribution, one can use the information about the correlations

between members in the chain. This is normally done with autocorrelation functions (from time series theory), that give the correlation between two components in the chain that are k iterations away from each other. That is, the kth order autocorrelation $R(k)$ is

$$R(k) = \frac{\sum_{i=1}^{N-k}(\theta_i - \hat{\theta})(\theta_{i+k} - \hat{\theta})}{\sum_{i=1}^{N}(\theta_i - \hat{\theta})^2} = \frac{Cov(\theta_i, \theta_{i+k})}{Var(\theta_i)} \tag{5.7}$$

where θ_i is the marginal distribution of parameter i in the MCMC chain. Usually low autocorrelation means fast convergence. In addition, one might want to plot $R(k)$ against k that should show geometric decay. Using autocorrelations in converge diagnostics is illustrated in figure 5.3.

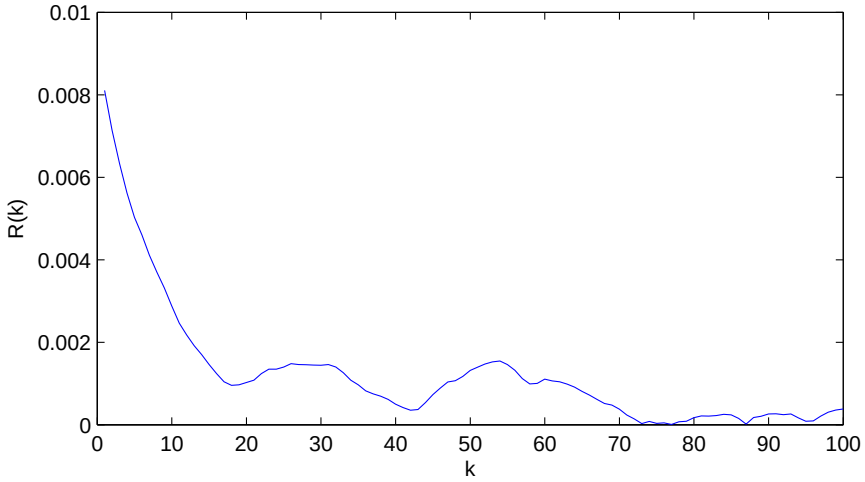

Figure 5.3: An autocorrelation plot for the first parameter in the kinetic model explained in section 9.2. The plot shows clear decay so the chain has probably converged when it comes to this parameter. In addition, if we take every 100th component from the chain, we can regard the thinned chain as a set of independent samples.

Another graphical method that appears in the literature is the CUSUM method. In this method the cumulative sums of the marginal distributions are calculated and plotted against the number of terms used in the sum, and the smoothness of the plot is investigated. [20]

In addition, some formal statistical tests have been developed to test the convergence of the MCMC algorithm. In the Geweke method, for example, the chain (burn-in discarded) is split into separate peaces and these are compared with a similarity test. [20]

A bit analogously with the Geweke method, one can calculate so called "batch means" from the chain. In this method, the precision of the sample mean is estimated by dividing the chain into small parts, "batches". The mean values of the batches are then calculated

and the variance of the means is estimated. When using this method, one has to make sure that the individual batches are long enough so the sample means are satisfactory estimates for the true means. In addition, one has to use a sufficient number of batches to get a good estimate for the variance. In literature, 20 different batches are suggested. [2]

5.5 MCMC Results and Visualization

The visualization of results is an important issue when making inference based on MCMC simulations. What kind of plots can we produce to present the new information, brought by MCMC, to the modeler? This question is discussed here.

Traditional regression analysis gives fixed values for unknown parameters and prediction curves. MCMC methods are based on random sampling and result in empirical distributions for unknown parameters. Moreover, it is possible to sample also values for model prediction at different points and construct a distribution also for the response curves of the model, that are called here *predictive distributions*.

Because MCMC results in empirical distributions, it is possible to derive information related to uncertainties in unknown parameters (identifiability) and in model predictions. In addition, pairwise examination of different parameters in the chain gives information about how the parameters correlate, which may lead to model refinement, for example.

5.5.1 Marginal Distributions

The MCMC methods produce samples from an n-dimensional posterior distribution of unknown parameters. The distribution of any subset of the parameter vector can be derived directly from the chain by choosing the values sampled for the parameters in question.

The most obvious plots that can be produced from the MCMC chain are the one-dimensional marginal distributions for each parameter. For estimating the density of the marginal distribution one can use some kind of non-parametric density estimation mechanism, for example histograms and kernel density estimators. The histogram approach for one-dimensional density estimation is illustrated in figure 5.4.

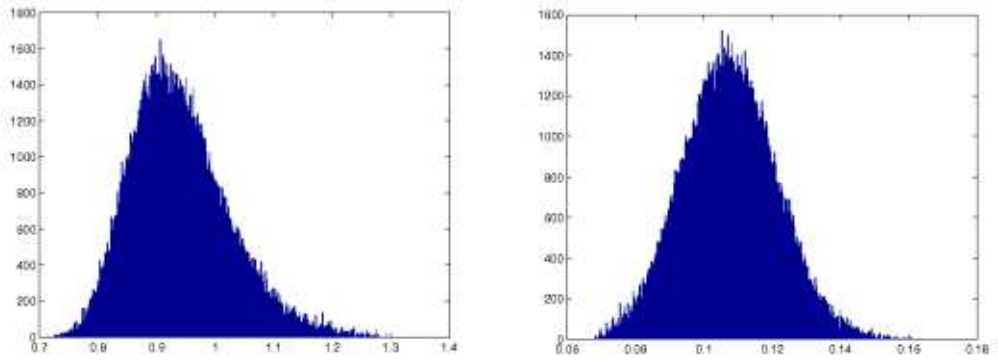

Figure 5.4: Histograms for unknown parameters from the model described in section 9.2. The drawback of using histograms is that many points are needed to produce a smooth plot. This plot was created with the Metropolis algorithm with chain length 250000.

One might also want to calculate some kind of confidence intervals for one-dimensional marginal distributions that approximately include some percentile $(1-\alpha)$ of the distribution mass. For samples of one parameter this is easy to do by sorting the sampled parameter values and approximating where the cumulative sum of the values reaches percentiles $\alpha/2$ and $(1-\alpha/2)$. The following table is created from the chain as in figure 5.4 with $\alpha = 0.05$. Table 1 shows the LSQ estimate, the empirical median of the chain and the empirical limits for the confidence interval.

Table 1: Least squares estimates and empirical confidence limits.

	LSQ	0.5	$\alpha/2$	$1-\alpha/2$
θ_1	0.9264	0.9308	0.8024	1.1311
θ_2	0.1080	0.1073	0.0815	0.1350

It is also common in MCMC analysis to plot the sampled points pairwise for every possible parameter pair, to reveal correlations between parameters (figure 5.5). In this case, one might want to construct a confidence region, that would approximately include a certain percentile of the two-dimensional marginal distribution mass. One could use the histogram approach in two dimensions as well by assigning a grid on the axes and see how many points fall into each box in the grid. Below some additional approaches for estimating the $(1-\alpha)$ level based on the sampled points are discussed.

One-dimensional marginal plots reveal problems in the identifiability of different parameters. The two-dimensional plots can reveal correlations and show that the ratio of two parameters is well identified, but the parameters themselves are not (a line in two-dimensional plots). For some models it may be useful to plot also three-dimensional

marginal distributions, which might reveal a plane-like behaviour between three parameters, that could not be visualized with lower dimensional plots.

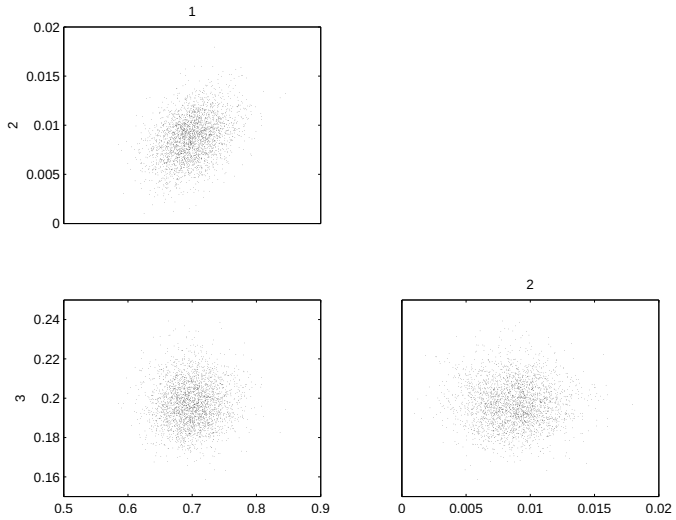

Figure 5.5: Sampled parameter values plotted pairwise for the example model described in section 9.2.

Kernel Density Estimation

For one-dimensional density plots and two-dimensional scatter plots the density in a desired point can be estimated using a sum of kernel functions at every data point. That is, in a one-dimensional case, in point x we estimate the density $d(x)$ with

$$d(x) = \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{5.8}$$

where the data points x_i are the sampled parameter values, in the MCMC context. Usually the kernel function K is the PDF of some well known distribution, like Gaussian. The parameter h is the "bandwidth", which describes the width (variance) of the kernel function - the bigger the parameter is, the smoother plots we get, since every point spreads wider to the neighborhood. Normally a rule of thumb (introduced in [21]) for h is used and we take

$$h = 1.06n^{-1/5} \min\left(s, iq/1.34\right) \tag{5.9}$$

where s is the standard deviation and iq the interquartile range (see Appendix 1) of the samples.

For the pairwise scatter plots the PDF of a bivariate Gaussian distribution $N(x_1, x_2, \sigma_1, \sigma_2, \rho)$ (for representation, see for example [8]) can be used as a kernel function, with some values given to variances (σ_1 and σ_2) and correlation ρ. In the bivariate case the variances

can be calculated using a rule of thumb as follows (see [21]).

$$\begin{aligned}\sigma_1 &= 1.06n^{-1/6}\min\left(s_1, iq_1/1.34\right)\\ \sigma_2 &= 1.06n^{-1/6}\min\left(s_2, iq_2/1.34\right)\end{aligned} \tag{5.10}$$

where s_1, s_2 (standard deviations), iq_1 and iq_2 (interquartile ranges) are calculated from the samples. For more on kernel density estimation, refer to [21].

The kernel density approach is illustrated in figure 5.6 for both pairwise scatter plot and one-dimensional marginal distributions. The kernel method is applied to one-dimensional parameter plots to get a smooth-looking PDF for the parameters. In two-dimensional density estimation, a grid is set on the axes and the density is estimated at each point.

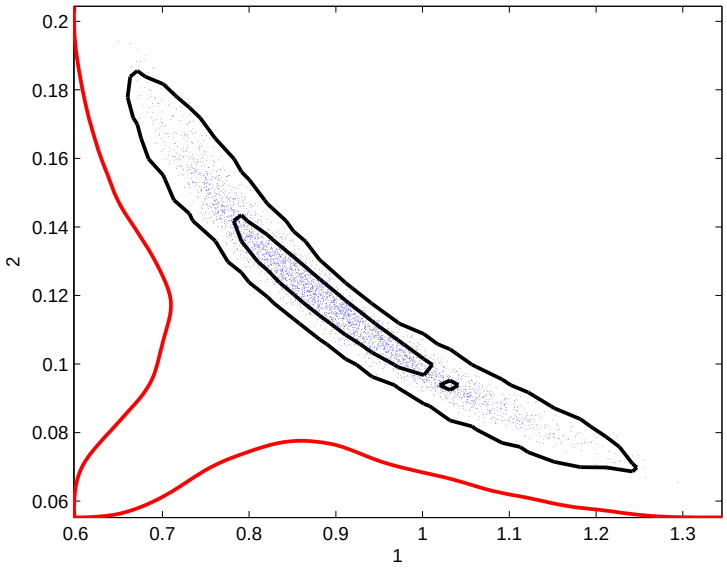

Figure 5.6: A scatter plot of a parameter pair with confidence regions based on kernel densities. 1D marginal distributions, also produced by the kernel method, are combined with the scatter plot.

Confidence Regions Using Gaussian Mixture Models

Another idea to form confidence regions is to use Gaussian Mixture models. A Gaussian Mixture model is a distribution with the PDF defined as a weighted sum of a number, say N_c, of Gaussian PDFs. That is

$$p(x|\mu,\Sigma) = \sum_{i=1}^{N_c} w_i p_i(x|\mu_i,\Sigma_i) \tag{5.11}$$

where $\sum_{i=1}^{c} w_i = 1$, which means that $\int p(\mathbf{x}|\mu, \Sigma)dx = 1$. That is, the model is formed from multivariate Gaussian "components" with means μ_i and covariances Σ_i.

The MCMC chains can be analyzed and represented in different ways using GMM. A Gaussian mixture model can be fitted to the produced chain. Especially, for plotting purposes, a number of bivariate Gaussian components can be fitted to the pairwise data of the chain. Then, the contours (ellipses) of a certain desired confidence level $(1 - \alpha)$ can be plotted. These can be graphically combined to produced an estimate of the overall $(1 - \alpha)$ confidence region. This **"combined GMM ellipses"** approach is illustrated in figure 5.7.

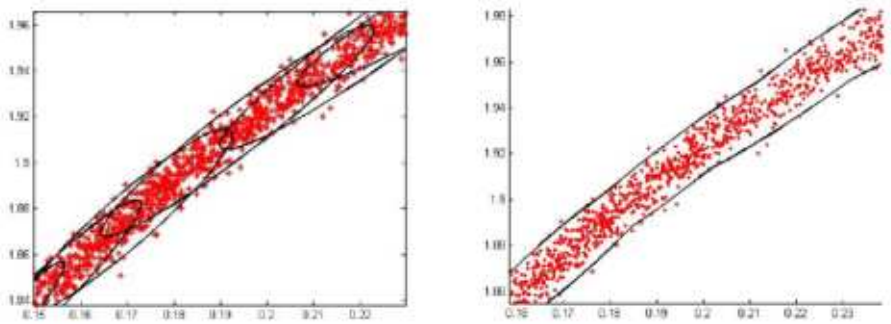

Figure 5.7: A scatter plot of a parameter pair with confidence regions based on a Gaussian Mixture Model. The confidence regions for the components of the mixture are first computed and the contours are combined.

The fitting of the GMM can be done, for example, using the Expectation-Maximization (EM) algorithm (see for example [6]). The fitting for the data set $\mathbf{x} = (x_1, ..., x_n)$ goes, in principle, as follows.

1. Generate an initial guess for the parameters $\mu_0 = (\mu_0^{(1)}, ..., \mu_0^{(c)})$ and $\Sigma_0 = (\Sigma_0^{(1)}, ..., \Sigma_0^{(c)})$ for c different components. Set $i = 0$.

2. (a) Calculate the posterior probabilities (weights) $w^{(j)}$ and normalized versions $\hat{w}^{(j)}$ for each component ($j = 1...c$) and for each point using the PDFs of the components

$$\begin{aligned} w^{(j)} &= p(x|\mu_i^{(j)}, \Sigma_i^{(j)}) \\ \hat{w}^{(j)} &= \frac{w^{(j)}}{\sum_{k=1}^{n} w_k^{(j)}} \end{aligned}$$

   (b) Calculate new means, covariances and weights for the Gaussian components

as expectations

$$\begin{aligned}
\mu_{i+1}^{(j)} &= \sum_{k=1}^{n} \hat{w}_k^{(j)} x_k \\
\Sigma_{i+1}^{(j)} &= \sum_{k=1}^{n} \hat{w}_k^{(j)} (x_k - \mu_{i+1}^{j})(x_k - \mu_{i+1}^{j})^T \\
w_{i+1}^{(j)} &= \frac{1}{n} \sum_{k=1}^{n} w_k^{(j)}
\end{aligned}$$

3. If the desired precision is achieved, stop. Otherwise set $i = i + 1$ and go to step 2.

A good initial guess for the parameters can be calculated, for example, using some clustering algorithm, such as K-means (see for example [22]).

The plotting mechanism based on the graphical combination of the confidence ellipses of the components of a fitted GMM is suitable for certain situations. Sometimes, however, the plots produced are angular: the points where two ellipses intersect produce sharp corners to the plots, as seen in figure 5.8. In addition, a lot of manual tuning is needed related to the number of fitted Gaussian components. Some ideas on how to automatically detect the "optimal" number of components exist, for example in [23].

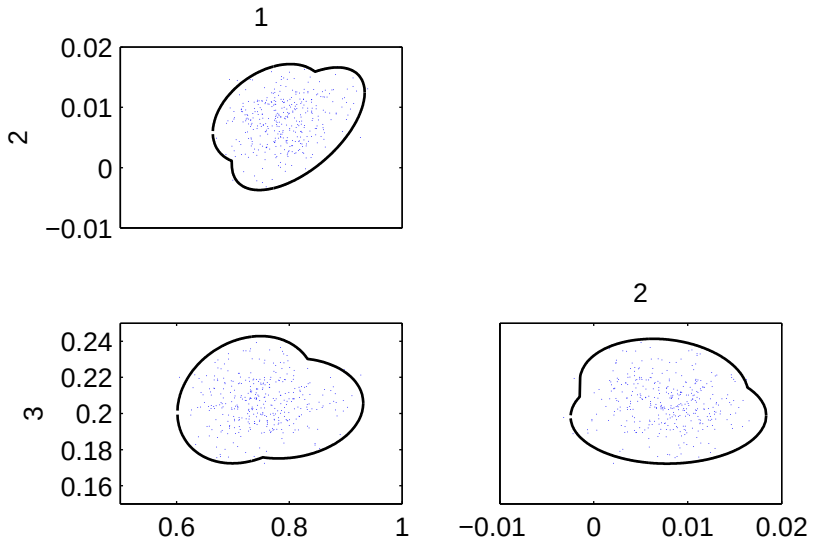

Figure 5.8: Scatter plots of parameter pairs with confidence regions based on a Gaussian Mixture Model. The points of intersection result in angular plots.

Another way to use the fitted Gaussian Mixture Model to two-dimensional density estimation and confidence region plots is to pick the direction of the Gaussian kernel function

(correlation ρ) in a certain point according to the component in the mixture model that the point belongs to. This **"GMM-directed kernel density estimation"** produces better density estimates and more accurate confidence regions in the pairwise scatter plots - especially in cases where the shape of the marginal distribution alters a lot.

In practice the directed kernel density estimation works as the kernel density estimation with a fixed covariance (variances and correlation in 2D case), with the exception that we now choose the correlation for the bivariate Gaussian kernel separately for every point using the covariance matrix of the corresponding GMM component. The GMM component that the point belongs to is decided by checking which weighted component gives the largest density (PDF value) to the point. That is, we choose the component for point x with

$$i^* = \max_i w_i p_i(x|\mu_i, \Sigma_i) \tag{5.12}$$

Example. Let us concider the two-dimensional version of a "banana shaped" target distribution, introduced in [24]. That is, the PDF of the target distribution is defined by $f_b = f \circ \phi_b(x)$, where f is the PDF of a bivariate Gaussian distribution $N(0, C)$ with $C = diag(100, 1)$ (contours are ellipses with one axis 10 times the size of the other). The twisting effect is brought to the target by defining $\phi_b(x) = (x_1, x_2 + b_{x_1^2} - 100b)$. The greater the parameter b is, the more twisted the target is. Here the value $b = 0.1$ is used. In figure 5.9 the directed approach is compared to the kernel density estimation with a fixed kernel, using 5000 points sampled from the target distribution with PDF f_b.

The computational complexity of the EM-algorithm is often quite high, especially when it comes to large data sets as is the case in analyzing MCMC output. Some new, faster versions of the EM-algorithm, designed especially for large datasets, have been proposed lately for example in [25]. These algorithms are to be tested for MCMC purposes in the future.

Due to the difficulties in the GMM approach for plotting confidence regions, the kernel density estimation approach with a fixed kernel function is used in this work as the basic way of forming confidence regions. However, if the kernel density approach does not work, the GMM-based approach can be tried. Gaussian mixture models are also useful in other aspects of MCMC analysis, some of which are listed below.

- Compression of the MCMC chain - a long chain can be expressed with few parameters (weights, means and covariances of the components)

- Producing new data - it is possible to sample new data from the mixture model, which is approximately distributed as the points sampled with MCMC
- New statistics from the chain - the correlations in the component Gaussians reveal the structure of the correlations in the model parameters

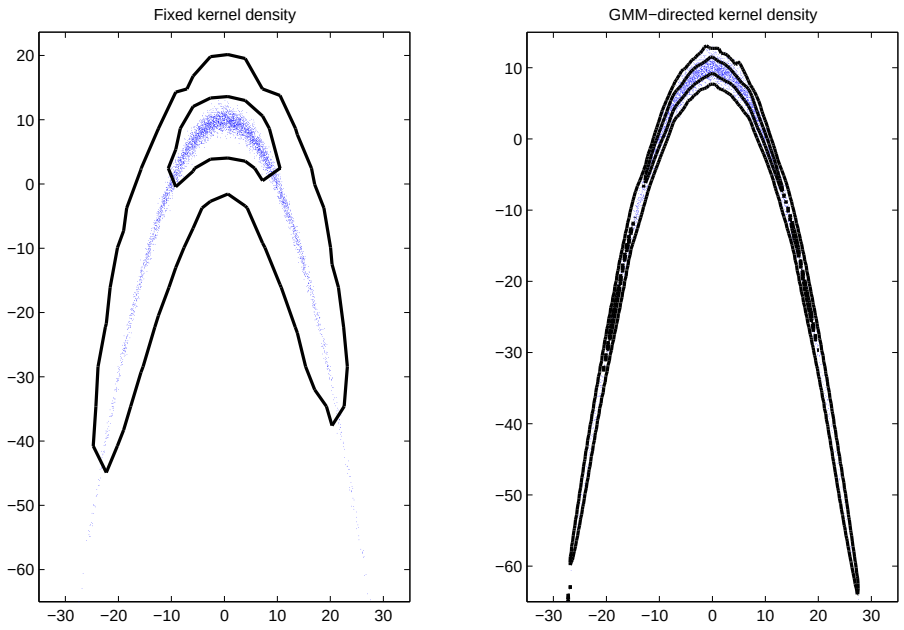

Figure 5.9: Density estimation of a twisted, thin distribution using 5000 sampled points. The left figure is produced by using a fixed kernel function, where ρ is taken as the correlation coefficient calculated from the points. The right figure is produced by choosing ρ separately for every point using a fitted Gaussian mixture model with 6 components.

5.5.2 Predictive Inference

The accuracy of the response of the model can also be evaluated through MCMC methods. That is, we are not limited the model parameters θ - we can, in fact, make inference about any function $f(\theta)$. The function can also be the model itself. If we calculate the model response values with different parameter values produced by MCMC methods, we get the so called $predictive\ distributions$. These model prediction curves and their distribution are, in fact, often more interesting than the distributions of the parameters.

In practice, this results in curves that represent the distributions of the model prediction and observations (see figure 5.10).

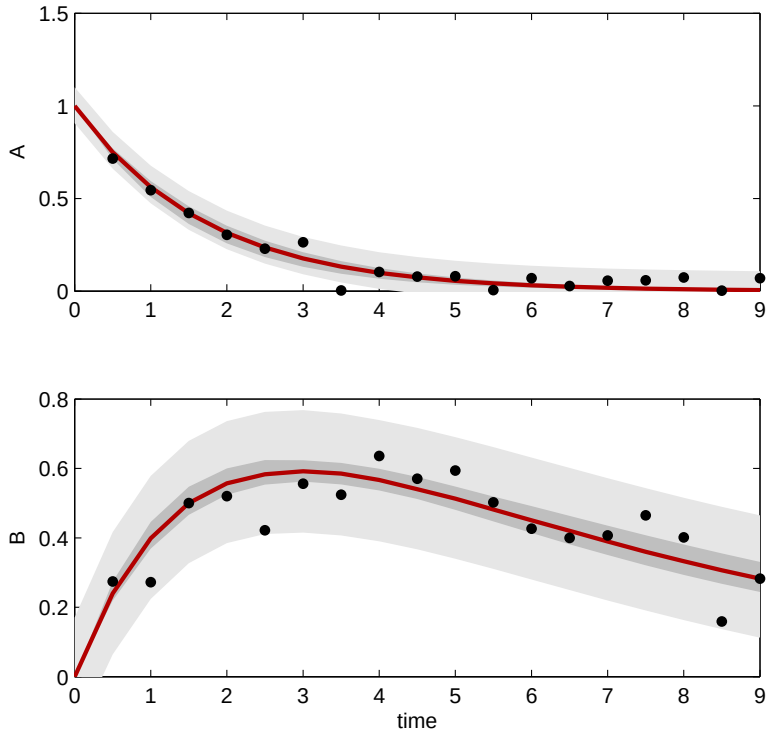

Figure 5.10: Predictive distribution for the response components for the example model described in section 9.2.

The confidence interval marked with darker gray in figure 5.10 is produced simply by calculating the response curves with the parameter values given in the sampled chain for given time points (the ODE systems that describe chemical reactions are time dependent), for example. This results in a sample of response values in different time points. That is, values for given time points are calculated using the model with the parameters in the MCMC chain. With the samples it is possible to create an empirical confidence interval for every time point. The $1 - \alpha$ confidence intervals for model prediction curves can be formed as follows.

1. Generate MCMC chain for unknown parameters
2. Calculate the model prediction for (a thinned part) the chain in order to create a chain for the model prediction.
3. Calculate the confidence interval for the prediction values separately for every time point
   (a) Sort the values

(b) Take the $\alpha/2$ and $1-\alpha/2$ empirical percentile of the samples (through interpolation)

4. Plot the limits for every (time) point

Of course the production of response samples does not have to be a separate procedure like presented above - the samples can be calculated in MCMC iterations after sampling new values to the chain. The response values could be saved, for example, in every 100th step.

The uncertainty of new observations can be modeled by adding "noise" to the calculated model prediction values. The added noise depends on the error variance (or covariance) of the measurements. The error can be assumed to be constant (some approximation) or it can also be sampled when the MCMC chain is constructed. The confidence intervals for the noisy predictions (lighter gray in figure 5.10) can be formed as stated above. The error sampling schemes are discussed further in section 5.6. In practice, the confidence area created by adding noise to the sampled prediction curves means, roughly speaking, the area from which observations (present and the ones yet to come) can be expected to found with a certain probability.

Different Ways to Model the Measurement Error

In figure 5.10 the noisy predictions, that form the lighter gray area, were formed by adding normally distributed noise ($\epsilon \sim N(0, \sigma^2 I)$) to the model predictions given by the parameters in the MCMC chain. That is, $y_{noisy} = y_{pred} + \epsilon$. When the prediction y_{pred} gets close to zero, as happens with response A in figure 5.10, the noisy predictions might get negative values and thus allow negative observations since $\epsilon > y_{pred}$. This happens, because the size of the measurement error is assumed to be constant between consecutive measurements, whereas in practice the measurement error is smaller when the values of the responses get small.

In figure 5.10 this problem is handled by cutting the negative noisy prediction area away from the plot and allowing only $y_{noisy} \geq 0$. An alternative way to model the noise structure is to calculate $\sqrt{y_{noisy}} = \sqrt{y_{pred}} + \epsilon$, which gives $y_{noisy} = (\sqrt{y_{pred}} + \epsilon)^2$ and thus all noisy predictions are positive. This magnifies the error a little, but is suitable for certain situations. When this kind of approach is used, the sum of squares in the likelihood in the MCMC run (equation 2.9) has to be calculated in the corresponding manner: here $SS_\theta = \sum_i (\sqrt{y_i} - \sqrt{f(x_i, \theta)})^2$.

A third way to add the error to the model predictions, so that the noisy predictions are all positive, is to assume that the noise is multiplicative and lognormal. That is, $\epsilon_{LOG} \sim Log - N(0, \sigma^2 I)$ which means that $\epsilon = \log(\epsilon_{LOG}) \sim N(0, \sigma^2 I)$. Now we can calculate $\log(y_{noisy}) = \log(y_{pred}) + \epsilon$ which gives $y_{noisy} = y_{pred} \exp(\epsilon) \geq 0$. In this case the likelihood is calculated using $SS_\theta = \sum_i (\log y_i - \log f(x_i, \theta))^2$.

In figure 5.11 the "squared error" and "multiplicative lognormal error" approaches are demonstrated. For the demonstrations, data was created for the squared error case with $Y_{data} = \left(\sqrt{Y_{model}} + \epsilon\right)^2$ and for the lognormal case with $Y_{data} = Y_{model} \exp(\epsilon)$. For the first component (A) the noise was created with $\epsilon_1 \sim N(0, 0.05)$ and for B with $\epsilon_2 \sim N(0, 0.1)$.

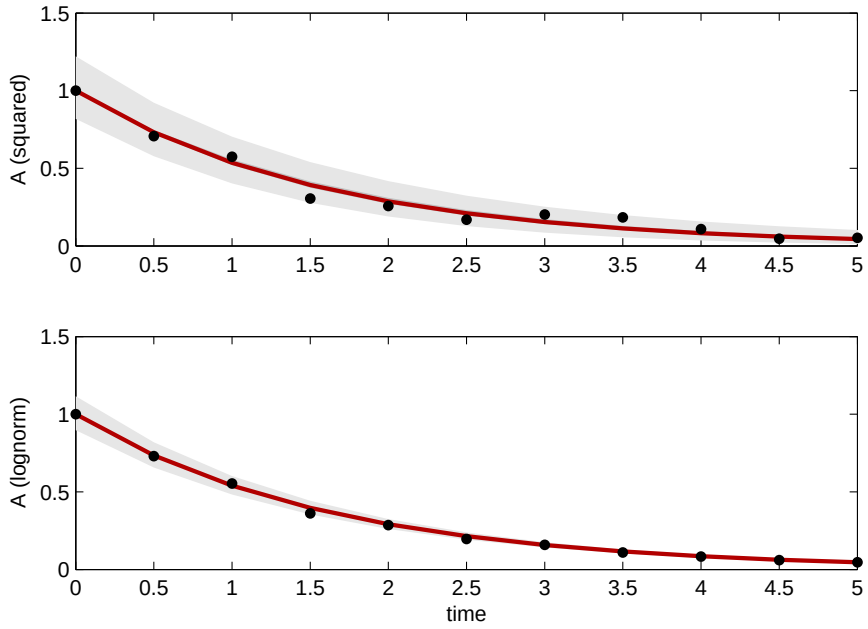

Figure 5.11: Predictive distribution plots for response component A of the model introduced in 9.2. The "squared error" approach is presented above and the "lognormal" case below. With these approaches no negative, "non-physical" predictions are simulated.

5.6 Sampling the Error Variance

In the algorithms presented in this work, the measurement error is modeled as $\epsilon \sim N(0, \sigma^2 I)$, and the Gaussian likelihood is constructed using this assumption. The error variance σ^2 is assumed to be known and it stays constant during the MCMC run. Note that the error variance might differ between different observed response components.

Normally, however, the error variance is not exactly known and an approximation has to be calculated using the measurements. If replicated measurements from observation points are available, the error variance can simply be estimated as the sample variance calculated from the replicated measurements. When this is not the case, the error variance can be approximated from the data using the classical formula (equation 5.6).

The error variance can also be regarded as a variable - we can let the variance "float" and sample it along with the unknown model parameters. In this section, one such sampling mechanism, based on a conjugate prior and Gibbs sampling, is presented. The presentation is based on [3].

We know, assuming a non-biased "perfect" model, that the error terms ($\epsilon_i = y_i - f(x_i;\theta)$) are distributed the same way as the error. If we assume that the error is normally distributed, we can write an expression for the conditional probability distribution of parameters and measurements with a given variance:

$$\begin{aligned} p((\theta,y)|\sigma^2) &\propto (\sigma^2)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\epsilon_i^2\right) \\ &= (\sigma^2)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}SS_\theta\right). \end{aligned}$$

We are interested in the posterior distribution of the error variance, $p(\sigma^2|(\theta,y))$. Using the Bayesian notation, we can write $p(\sigma^2|(\theta,y)) \propto p(\sigma^2)p((\theta,y)|\sigma^2)$ where $p(\sigma^2)$ represents the prior density for the variance.

How can we assign the prior for the variance so that the posterior distribution is some known distribution, from which we can produce samples efficiently? If we use conjugate priors (section 2.3.1), the posterior has the same form as the prior. For the normal distribution with unknown variance, the conjugate prior is the *inverse Gamma* distribution, which can be rewritten (with different parameterization) to the form of the *scaled inverse* χ^2 distribution (Inv-$\chi^2(n,S^2)$), that has the following PDF form with n degrees of freedom and scale parameter S^2:

$$p(x) \propto x^{-(n/2+1)}e^{-(nS^2)/2x} \tag{5.13}$$

which is equivalent to the *inverse Gamma* distribution ($\Gamma^{-1}(\alpha,\beta)$) with parameters α and β, that has the PDF

$$p(x) \propto x^{-(\alpha+1)}e^{-\beta/x}. \tag{5.14}$$

It can be easily seen, that the likelihood function above assumes the form of the PDF of the Inv-$\chi^2(n, S^2)$ distribution (and thus inverse Gamma), if σ^2 is regarded as the variable and θ constant. That is, when we multiply the prior and the likelihood, they get a similar form, with different parameters.

Now we assign a conjugate prior to the variance from the scaled inverse-χ^2 (inverse Gamma) as $\sigma^2 \sim Inv - \chi^2(n_0, \sigma_0^2)$. The resulting posterior density is

$$\begin{aligned} p(\sigma^2|(\theta, y)) &\propto p(\sigma^2)p((\theta, y)|\sigma^2) \\ &\propto \left(\sigma^2\right)^{-(n_0/2+1)} \exp\left(\frac{-n_0 S_0^2}{2\sigma^2}\right) \left(\sigma^2\right)^{-n/2} \exp\left(\frac{-SS_\theta}{2\sigma^2}\right) \\ &= \left(\sigma^2\right)^{-((n_0+n)/2+1)} \exp\left(-\frac{1}{2\sigma^2}\left(n_0 S_0^2 + SS_\theta\right)\right). \end{aligned}$$

That is, using the inverse Gamma, we can write the posterior distribution for the variance as

$$\sigma^2|(\theta, y) \sim \Gamma^{-1}\left(\frac{n_0 + n}{2}, \frac{n_0 S_0^2 + SS_\theta}{2}\right) = \Gamma^{-1}(a, b). \tag{5.15}$$

In practice, the Gamma distribution is more convenient for sampling, since many scientific computing environments include the functions to sample from it. We can use the relationship between Gamma and Inverse Gamma distributions and calculate the variance with $\sigma^2 = 1/\gamma$ where $\gamma \sim \Gamma(a, b^{-1})$. This can be done, because if $X \sim \Gamma^{-1}(a, b)$ then $Y = 1/X \sim \Gamma(a, 1/b)$.

When the result is coupled with a MCMC sampling algorithm for the parameters, we arrive at a Gibbs sampling procedure (see section 4.5). That is, we assign some initial value for the model parameters (usually the LSQ optimum), and then sample the variance from the conditional distribution as described above. In the next MCMC step we use the sampled variance and thus calculate a new θ conditional on σ^2, which we use to sample a new σ^2. The sampling procedure adds only one step to the MCMC algorithms, after the acceptance step.

The parameter n_0 in the prior distribution can be thought to represent the number of observations equivalent to the information given by the prior. The S_0^2 represents the average squared deviation of the observations [3]. In practice, $n_0 = 1$ and $S_0^2 = MSE$ (see equation 5.6) are often chosen in this work. The parameter n_0 is often chosen to be small (such as 1 or 0.01 or 0.001), which is an attempt to make the prior uninformative, as explained in [26]. If we choose $n_0 = 0$, the effect of S_0^2 vanishes, but since SS_θ can be close to 0, we often need some kind of prior and prefer a small number for n_0. On the other hand, also

informative versions of the prior can be used, if we have some kind of a priori knowledge about the behavior of σ^2.

If the error is modeled to be multivariate Gaussian with some covariance matrix, the conjugate prior distribution for the covariance is Inverse-Wishart. That is, also the error covariance matrix can be sampled at each MCMC step, if we do not want to make the assumption $C = \sigma^2 I$. In this case we allow the different response components to have correlated errors. For more information about error modeling, see [3] and [27].

6 Adaptive MCMC Algorithms

In the basic MCMC method based on random walk Metropolis-Hastings (see section 4.2) the problem is how to choose the proposal distribution so that the algorithm converges as fast as possible. This normally requires a lot of manual tuning of the proposal. When using a Gaussian proposal, the problem is to find a suitable covariance matrix for the proposal.

Adaptive MCMC methods use the history of the iterative process (the chain created so far) to update the proposal distribution during the computation. Several algorithms are recently introduced, using a bit different adaptation schemes to update the covariance matrix of a Gaussian proposal distribution. The Adaptive Proposal (AP) and Adaptive Metropolis (AM) algorithms are discussed in the next section. After that, some further methods, called DRAM (Delayed Rejection Adaptive Metropolis) and SCAM (Single Component Adaptive Metropolis) are presented. No convergence and ergodicity investigations are made here, for them refer to the original papers ([28],[18]). The goal is to present the algorithms in a practical form from which they are straightforward to implement.

It is clear that the adaptive algorithms lead to a stochastic process that is not Markovian, because the dependence on the history reaches further back than to the previous state. For some of the methods, however, it can be shown that the algorithm has appropriate ergodicity properties so that it generates samples correctly from the invariant target distribution.

The adaptive algorithms mentioned above are based on updating the covariance matrix of a Gaussian proposal distribution. That is, both the width and the orientation of the proposal distribution are updated iteratively.

6.1 Adaptive Proposal and Adaptive Metropolis

The Adaptive Proposal (AP) and Adaptive Metropolis (AM) algorithms were introduced in [24] and [29]. The only difference is that in AP the covariance of the Gaussian proposal distribution is calculated using a fixed number of previous states, whereas in AM an increasing part of the whole chain calculated so far is used in the adaptation. Both of the algorithms are based on updating the covariance of a Gaussian proposal, as calculated from the sampled chain. A danger of adaptive schemes is that they may lead to incorrect convergence, since the process is not Markovian anymore and the standard ergodicity re-

sults do not apply. Indeed, this is the situation with AP, while the authors could prove the ergodicity of the AM algorithm ([29]).

The adaptive methods simply add one step to the simulation loop of the basic MCMC algorithm. If we look for example the algorithm presented in section 4.2, we can add an "adaptation step" to the algorithm after the "acceptance step" (the if-structure):

- Do Adaptation
  - Calculate C_{t+1}
  - Calculate R so that $C_{t+1} = R^T R$

How is the new covariance matrix C_{t+1} calculated? Let us suppose here that we are at time t in the algorithm and we already have created chain $(X_0, X_1, ..., X_t)$. The proposal distribution is now Gaussian with mean at the current state X_t and covariance C_{t+1}. For the AP algorithm we use a fixed history length, say l, and put

$$C_{t+1} = s_d Cov(X_{t-l}, ..., X_t), \quad when\, t > t_0. \tag{6.1}$$

In the AM algorithm we use the whole history of the chain, and the adaptation rule can be defined as

$$C_{t+1} = s_d Cov(X_0, ..., X_t), \quad when\, t > t_0 \tag{6.2}$$

where t_0 is the initial period after which the adaptation is begun. When $t < t_0$ we can use a fixed initial covariance C_0. The constant s_d is a scaling parameter that depends only on the dimension of the parameter space. In practical implementations we use a rule of thumb for the scaling parameter, introduced in [30]: $s_d = 2.4^2/d$. This can be shown to optimize (in some sense) the mixing properties of the random Metropolis walk when using Gaussian target and proposal distributions. [29]

The adaptation process can be "thinned", which means that the adaptation is done after a certain period - during a number of steps we use a fixed, previously adapter proposal, after which we do adaptation again. The covariance is calculated using the formula for the empirical covariance matrix (see Appendix 1). This, however, requires a lot of computation if we do it every time we perform adaptation. In chapter 6.1.4 some ways to update the covariance recursively, based on the values from the previous adaptation step, are presented.

The Adaptive Metropolis algorithm is compared to the MH algorithm in the next example.

Example. Let us consider a situation, where we have unknown parameters $\theta = [a, b]$ and model response y and we have to form the distribution of the parameters. We define the likelihood for the parameters so that the parameter a has to be close to y. In addition, we know a priori that the parameter b has to be very close to a^2. That is, we define the likelihood as

$$p(y|(a,b)) = \exp\left(-100(b-a^2)^2\right)\exp\left(-(a-y)^2\right) = \exp\left(-100(b-a^2)^2 - (a-y)^2\right).$$

The negative log-likelihood function $-\ln(p(y|(a,b))) = 100(b-a^2)^2 + (a-y)^2$ reminds the *Rosenbrock* function, which is often used as a benchmark function for different optimization routines. That is, the distribution of the parameters here follows the Rosenbrock function, which has a rather difficult shape, which makes the tuning of the proposal distribution for MCMC sampling more difficult.

In the example we set $y = 1$. Both normal Metropolis MCMC and AM are run with different Gaussian proposal distributions with mean at the current point. A fixed covariance of form $C_0 = \frac{\sigma^2}{k}I$ is used for the Metropolis MCMC and for the AM before the first adaptation. In figure 6.1, several runs of length 5000 are run, with different values for the scaling parameter k. By altering k we change the size of the fixed Gaussian proposal distribution. The acceptance rate and the scatter plots are examined to make conclusions about the performance of the different approaches.

From figure 6.1 one can see that the performance of Metropolis algorithm is very sensitive to the proposal covariance. With AM we get reasonable results and better coverage of the posterior distribution, even if the initial covariance was not too well chosen. If the initial proposal is too large, AM can run into problems related to singularity - the covariance is calculated from a set of replicates of a single points. In the cases where the algorithm does not move in the beginning, we might try a larger adaptation interval and scale the initial proposal covariance so that it becomes smaller. Some tricks to overcome difficulties in AM are discussed in section 6.1.2. In section 6.3 the DRAM algorithm, that helps to get the sampler "moving", is discussed.

The AM algorithm seems to work especially well with strongly correlated distributions. This seems natural, since the proposal distribution adapts to the rapidly changing behavior of the posterior. The algorithm has been tested to work up to 200-dimensional problems ([29]). When the dimension gets higher, the AM requires increasingly long simulations. Another adaptive method, designed especially for high dimensional problems, is introduced in the section 6.2.

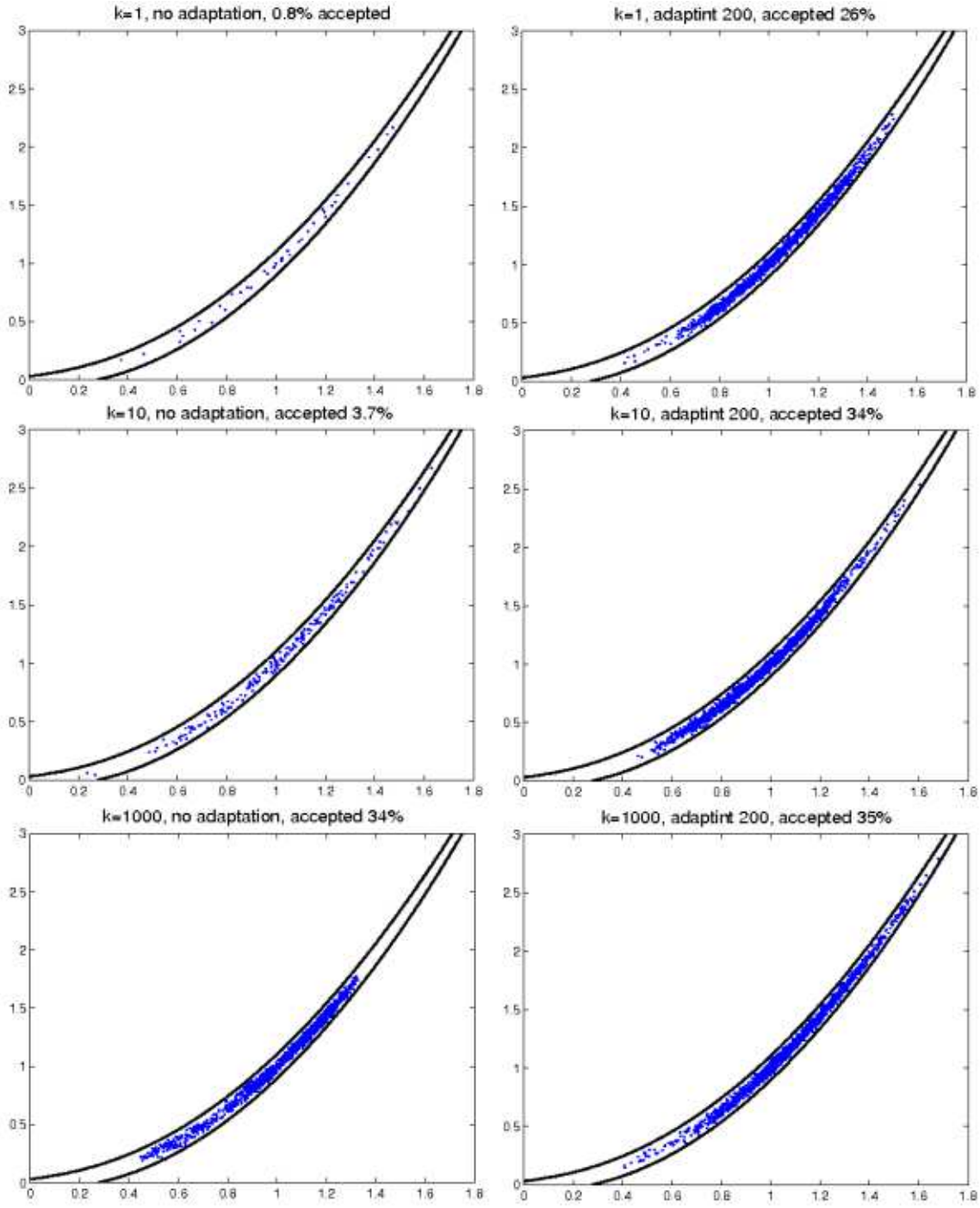

Figure 6.1: The scatter plots in the left column were produced by Metropolis MCMC and the right ones with AM. Here $\sigma^2 = 0.1$. Adaptation interval for the right column was 200 and scaling parameter was 1, 10 and 1000 for each row respectively. The dark black contour line represents the area where the posterior probability (Rosenbrock function) is high.

6.1.1 Adaptation Interval

How should the adaptation in AM be done? One must make sure that the chain produces enough diverse points during the adaptation interval so that the covariance matrix can be calculated. This is especially important in the beginning. Because of problems related to singularity the adaptation interval is normally taken to be quite large. In general, the more the chain moves in the beginning, the smaller the adaptation interval can be. Adaptation intervals ranging from 50 to 1000 are used in the empirical part of this work.

The effect of cutting down the adaptation interval can be normally seen in better mixing and in higher acceptance ratio. It depends on the case, however, how strong the effect of adaptation is. To illustrate the effect, the development of acceptance ratio in the AM run is plotted in figure 6.2 for 3 example models discussed in chapter 9. One can see, that the difference compared to standard MH depends strongly on the case.

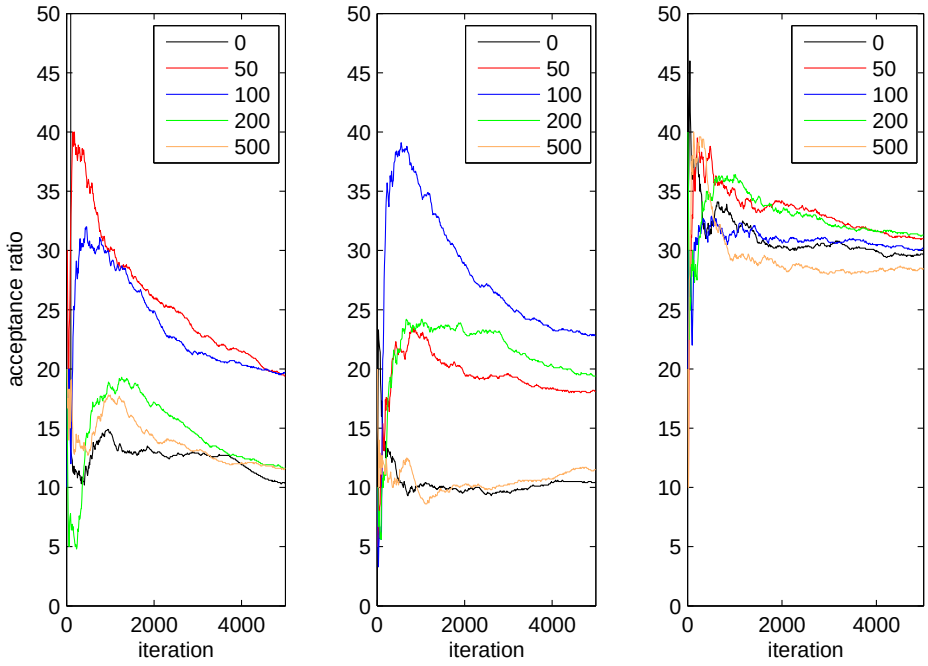

Figure 6.2: The development of acceptance ratio with different adaptation intervals for 3 models found from chapter 9. For the first model from the left, the adaptation intervals 50 and 100 seem to provide best results. For the second model, the adaptation interval 100 seems to result in higher acceptance ratio, whereas the efficiency of the plain MH is quite poor. In the third picture, the adaptation does not have a significant effect on the acceptance ratio.

6.1.2 Greedy Start

A pitfall in using Adaptive Metropolis is the possible slow start of the algorithm. If the algorithm stays still for a long time in the beginning of the run, the covariance calculated from the sampled points will become singular and the algorithm will fail. This is why the adaptation interval is normally chosen to be quite large - there has to be some movement in the beginning in order to get a reasonable covariance matrix. For later iterations a smaller adaptation interval could be used (the adaptation could be done more frequently). One idea would be to alter the adaptation interval as the algorithm proceeds, so that in the later iterations the adaptation would be done more often. In addition, a separate larger adaptation interval can be used for the burn-in period.

One way to make sure that the covariance matrix will not become singular is to apply a *greedy start* procedure. In greedy start, the proposal distribution (covariance matrix) is updated using only the accepted states for a short initial period. That is, all the points from which the covariance is calculated differ from each other. [29]

If the algorithm does not move or the acceptance ratio is too low, it is because the proposal distribution is too wide. One way to speed up the algorithm is to "shrink" the proposal distribution by multiplying the width parameter (variance / covariance in Gaussian proposals) by some scaling factor. On the contrary, if the acceptance ratio is too high, scaling upwards works as well. [29]

6.1.3 Initial Covariance

An additional implementation issue related to AM is the role of the initial covariance of the proposal distribution. In the initialization step some kind of initial covariance is formed, typically based on MSE and linearization as explained in chapter 5.2, for example. When the first adaptation is calculated, how is the initial covariance taken into account?

One possibility is to totally discard the initial covariance and merely calculate the first adapted covariance from the sampled points. Another, perhaps more efficient way is to take the initially fitted covariance into account as well. This can be done, for example, by interpreting the initial covariance as if it had been calculated from a number of points, say n points, after which the first covariance C_1 can be calculated using the recursive

formula. Basically this means that with different choices for n the initial covariance can be weighted - if large n is chosen, we trust the initial covariance more. In addition, several AM chains may be calculated, with initial covariance taken as the final covariance of the previous run. [29]

6.1.4 Updating the Proposal Covariance

In the adaptation step, the new proposal covariance C_{t+k} (where k is the adaptation interval) can be calculated simply by applying the empirical covariance formula (appendix 1) to the whole chain $(X_0, ..., X_{t+k-1})$. However, when the chain lengths get very long, the calculation of the covariance becomes a large computational burden. Here we discuss the option of updating the covariance recursively, using the covariance, mean and sampled points calculated during the last adaptation interval. That is, we try to find the function f so that $C_{t+k} = f(C_t, \overline{X}_t, X_{New})$ where $\overline{X}_t$ is the mean of $(X_0, ..., X_{t-1})$ and $X_{new} = (X_t, X_{t+1}, ..., X_{t+k-1})$ includes the new sampled points.

Let us consider a situation where we have sampled points $(X_0, ..., X_t)$. Then, as stated in [29], using the formula for the empirical covariance matrix, we can write the next covariance C_{t+1} using the previous covariance $C_t = Cov(X_0, ..., X_{t-1})$ and the new sampled point X_t as follows:

$$C_{t+1} = \frac{t-1}{t} C_t + \frac{s_d}{t}(t\overline{X_{t-1}X_{t-1}}^T - (t+1)\overline{X}_t\overline{X}_t^T + X_t X_t^T). \tag{6.3}$$

The mean $\overline{X}_t$ can also be calculated recursively using the previous mean $\overline{X}_{t-1}$. The formula for updating the mean is derived below.

$$\begin{aligned} \overline{X}_t &= \frac{X_0 + X_1 + ... + X_t}{t+1} \\ &= \frac{t}{t+1}\frac{X_0 + X_1 + ... + X_t}{t} + \frac{X_t}{t+1} \\ &= \frac{t}{t+1}\overline{X}_{t-1} + \frac{X_t}{t+1} \\ &= \overline{X}_{t-1} + \frac{t}{t+1}(X_t - \overline{X}_{t-1}). \end{aligned} \tag{6.4}$$

Finally, the recursive formula for calculating C_{t+k}, found by continuing the one-step iteration (equation 6.3), is given as

$$C_{t+k} = \frac{t-1}{t+k-1}C_t + s_d(\frac{t}{t+k-1}\overline{X}_{t-1}\overline{X}_{t-1}^T - \frac{t+k}{t+k-1}\overline{X}_{t+k-1}\overline{X}_{t+k-1}^T + \frac{1}{t+k-1}X_{new}X_{new}^T). \quad (6.5)$$

In addition, a formula for the k-step update for the mean, $\overline{X}_{t+k-1}$, that is needed in the k-step covariance update above, can be written by extending the one-step mean update formula (equation 6.4):

$$\overline{X}_{t+k-1} = \frac{t}{t+k}\overline{X}_{t-1} + \frac{1}{t+k}\sum_{i=t}^{t+k-1} X_i. \quad (6.6)$$

The covariance update using the k-step formula (equation 6.5) directly results in shorter and possibly faster code implementations. However, this approach might lead to numerical difficulties such as inaccuracies and number overflows, since the formula contains matrix multiplications where the cells might have very large values. Thus, in practice, it is safe and advisable to use the one-step update formula (equation 6.3) iteratively k times to produce C_{t+k} from C_t.

Formulas (6.3) and (6.4) also show how the AM algorithm approaches MH when $t \to \infty$, since $C_{t+1} \to C_t$ and $\overline{X}_t \to \overline{X}_{t-1}$. That is, the proposal converges towards a fixed distribution. The proof of ergodicity of the AM algorithm is also based on this fact (see [29]).

6.2 Single Component Adaptive Metropolis

The idea of adaptation can also be applied to the SC algorithm presented in section 4.4. In Single Component Adaptive Metropolis (SCAM), the one-dimensional Gaussian proposal distributions used in SC are adapted individually as in AM. The algorithm is similar to SC, with the exception that for each component X_t^i the candidate point is generated from distribution $N(X_{t-1}^i, c_t^i)$ where variance c_t^i is updated for each component using the adaptation rule

$$c_t^i = sVar(X_0^i, ..., X_{t-1}^i). \quad (6.7)$$

As in AM, the adaptation is only applied after an initial period, before which the variances for the conditional proposals for components are kept fixed. The scaling parameter s is in practical implementations chosen as in AM and is therefore $s = 2.4$ (dimension $d = 1$). Again, to avoid computational costs, a recursive formula for updating the variances can be

formulated. [18] The SCAM algorithm performs well in high-dimensional cases where there is no strong correlation between the parameters. It seems to work well also in some cases with strong correlation, but not in always. In general, truly high-dimensional problems still require more research.

6.3 Delayed Rejection Adaptive Metropolis

The Delayed Rejection algorithm (DR), introduced in [28], is a modification of the standard Metropolis-Hastings algorithm that has been proved to improve the efficiency of MCMC estimators. The idea in DR is that in case of rejection in the acceptance step we propose another move instead of storing the old parameter values in the chain. The acceptance probability of this "second stage" acceptance step is chosen so that the reversibility conditions of the chain are preserved and thus the chain stays ergodic. The second stage move depends on the current position and on the point that has been rejected in previous stage. The delayed rejection mechanism can be extended to any number of stages. More on DR can be found from [31] and [32].

Delayed Rejection Adaptive Metropolis (DRAM) combines adaptation to the DR procedure. Here, after every AM step done with an adapted covariance C_t, DR is applied upon rejection so that for stage i the proposal covariance $C = C_t^i$. The covariance at DR stage i can be computed for example simply by scaling the covariance produced by the AM-step: $C_t^i = \gamma_i C_t$, where $i = 1...m$. Here m is the number of DR stages applied for every rejected point. The purpose of the algorithm is to guarantee that at least one of the proposals is chosen sufficiently. Other second stage moves can be designed as well.

The DRAM algorithm improves the efficiency compared to standard MCMC and AM approaches especially, when the initial point is badly chosen and the parameters are not well identifiable. In addition, if the algorithms have difficulties in getting themselves moving (the acceptance ratio is very low), the DRAM algorithm, with second stage moves scaled down, can provide help. In easier cases, the algorithm does not perform significantly better than AM, for example.

In the empirical part of the work, for most of the models, the AM algorithm was able to produce a sufficient sample from the posterior. In one case, however, the acceptance ratio stayed very low with AM, whereas DRAM seemed to produce a more comprehensive sample from the posterior (see section 9.6).

7 Population Monte Carlo

In MCMC sampling algorithms presented in chapters 4 and 6, one point from the posterior distribution is sampled at each iteration. The convergence of the algorithms is ensured through the ergodicity theorems. In addition, no general stopping rules can be formed and the convergence is assured by different visual methods.

In Population Monte Carlo (PMC) methods we produce, at each iteration, a sample of size n from the target (posterior) distribution π. In addition, every sample created gives an unbiased estimator of the mean of the target. That is, the algorithm can be stopped at any time, only the accuracy of the estimator improves along the iterations. The user does not have to worry about either dependencies that are due to the Markovian nature of MCMC or convergence of the algorithm. PMC can also, if well designed, be a little faster than standard MCMC algorithms.

The PMC method that has been under research recently, suitable for sampling from the posterior of the Bayesian framework, is based on iterative replication of the importance sampling procedure discussed in section 3.2.4. The background of the method is in the family of *Sequential Monte Carlo* (SMC) methods, also referred to as *Particle Filters*, that are primarily designed for dynamical models, that can be described as a discrete time series by

$$x_{t+1} = f(x_t, v_t) = f(x_t) + v_t \tag{7.1}$$

$$y_t = h(x_t, e_t) = h(x_t) + e_t \tag{7.2}$$

where v_t and e_t are noise signals. In the second equations, the additive noise assumption is made.

The most simple iterative importance sampling methods used for dynamical models are the Sequential Importance Re-sampling (SIR) and the Sequential Importance Sampling algorithms. More about SMC methods for dynamical models can be found from [33], [14], [34] and [35].

The methods can, however, be modified to work also in the case of a static target distribution and static unknown parameter values, as is the case in the formulation of the problems in this work. Especially PMC methods are developed to utilize iterative importance sampling in parameter estimation of static models.

The purpose of this chapter is to introduce the backgrounds for the PMC approach and present a simple algorithm for the method. Since this work concentrates mainly on MCMC, the PMC method is just presented as a promising alternative for tasks that need sampling. The approach is theoretical and only a simple example is used to demonstrate PMC in practice. The theoretical investigation related to PMC is based mainly on [36], [37] and [38].

7.1 PMC algorithm

The PMC method is quite close to the SIR method - actually PMC can be thought as a re-formulation of SIR for static models. In literature the PMC is often described as a competitor to MCMC methods. However, the PMC and MCMC methods are not that different in nature - they both are based on sampling from a "wrong" distribution; in MCMC we sample from the proposal distribution and in PMC we use the importance function. In addition, MCMC steps can be used as a part of the PMC algorithm (and vice versa) as discussed later in this section.

Let us consider a sample $x^{(t)} = (x_1^{(t)}, ..., x_n^{(t)})$ that is created at iteration t. In MCMC, the next sample $x^{(t+1)}$ would be created from a proposal $q(.|x^{(t)})$. The usage of the "wrong" distribution is justified with the ergodic theorem. In PMC we, instead, make a correction to the "wrong" distribution at each step using importance weighting (see chapter 3.2.4 for details).

If we have, at iteration t, sampled the components of $x^{(t)}$ from the importance distributions $g_{it}(x_i^{(t)})$, the importance weights can be defined as

$$w_i^{(t)} = \frac{\pi(x_i^{(t)})}{g_{it}(x_i^{(t)})} \tag{7.3}$$

where $i = 1, ..., n$. That is, in PMC the importance function may depend on the index of the sample i and iteration index t, a bit analogously with Single Component MH and Gibbs Sampling. We can, of course, define a "global" importance function $g_t = g_{it}$, that would be analogous with the proposal distribution in the Metropolis-Hastings algorithm presented in chapter 4.2. It can be shown that the dependency on both i and t does not affect the validity of the importance sampling scheme.

Since the posterior distribution is only known up to a normalizing constant ($\pi(x) =$

$\pi(x|y) = Kf(y|x)\pi_{pr}(x)$), we get unnormalized weights by using $\pi(x) = f(y|x)\pi_{pr}(x)$ in equation (7.3). In the PMC algorithm we need the normalized weights $\hat{w}_i^{(t)}$ so that $\sum_{i=1}^{n} \hat{w}_i^{(t)} = 1$, which can be done with

$$\hat{w}_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i=1}^{n} w_i^{(t)}}. \tag{7.4}$$

Now, the PMC algorithm for producing a sample $X^{(t)} = (X_1^{(t)}, ..., X_n^{(t)})$ can be written in pseudo-code as follows [6].

1. For $t = 1, ..., N$
   (a) For $i = 1, ..., n$
      - Select importance function g_{it}
      - Generate $x_i^{(t)} \sim g_{it}$
      - Calculate $w_i^{(t)}$ using (7.3)
   (b) Endfor
   (c) Calculate $\hat{w}_i^{(t)}$ using (7.4)
   (d) Re-sample n samples from $x^{(t)}$ with replacement according to $\hat{w}_i^{(t)}$ to produce $X^{(t)}$.

In the re-sampling step, a re-sampling with replacement is conducted. Sampling with replacement here means basically the same as in Bootstrap methods (section 3.1.3) for creating new samples from existing data: random indexes are chosen. The previous indexes are replaced with the created ones, that can contain replicas of different points. In the PMC algorithm, the re-sampling is done according to the weights that different particles (sampled points) have. That is, particles with large weights are replicated and moved to the next generation of particles more likely than the particles with small weights. The approach is a bit analogous with the selection process in evolutionary algorithms. In practice, the re-sampling can be done (for example) by constructing an empirical CDF from the weights and using the inverse CDF method (see section 3.1.1).

Here we note that the importance functions calculated at each step may depend on previous importance samples. That is, the algorithm can be described as an adaptive iterated importance sampling. It can be shown, however, that the adaptation of the importance

function does not affect the correctness of the algorithm. That is, adaptive schemes for updating the importance weights can be designed with less restrictions. In the case of adaptive MCMC algorithms, thorough ergodicity calculations have to be carried through in order to justify the method that is no longer Markovian. This kind of flexibility in importance function design can also be calculated as one of the advantages of PMC when compared with MCMC.

How can the importance functions g_{it} be formed? One idea is to use some distributions that are easy to sample from and that are centered at the previous point $X^{(t-1)}$, for example Gaussian distributions. If separate importance functions are defined for each component, the importance function g_{it} can be for example a normal distribution $N(X_i^{(t-1)}, \sigma_i^{(t)})$ where $\sigma_i^{(t)}$ is calculated from the previously sampled points $(X_i^{(1)}, ..., X_i^{(t-1)})$. In addition, different scales for the importance distribution can be chosen according to the performance of previous proposal distributions $q_{i(t-1)}$. The performance measure for a certain importance function can be, for example, the number of re-sampled points that differ from each other. On the other hand, if a global importance function $g_t = g_{it}$ is used, the importance function can be, for example, a multivariate Gaussian distribution $N(x^{(t)}, C^{(t)})$ where covariance $C^{(t)}$ is calculated from the previously created sample or from the whole past. The former approach is analogous with the SCAM method and the latter with the AP and AM methods from the adaptive MCMC methodology presented in chapter 6. The SCAM-style PMC is, at least intuitively, promising also in high-dimensional scenarios.

One problem of the re-sampling step in the algorithm is that the sample produced by re-sampling with replacement might contain many replications of the same point. One idea to increase the diversity of the created sample, introduced in [39], is to use a standard MCMC step after the re-sampling. That is, new sample $Y^{(t)}$ is produced from $X^{(t)}$ in every iteration with a simple MCMC step to make the points differ from each other. In this way the PMC and MCMC methods can be coupled.

The simplest way to utilize the PMC idea in standard MCMC algorithms with Gaussian proposal distributions is to make the center point of the importance distribution move with MCMC using standard Metropolis-Hastings acceptance rules, and at each step produce n samples from the importance distribution with the resampling step. That is, the location of the importance distribution is defined by the MCMC method, but the sampling itself relies on PMC. This kind of approach leads to **PMCMC** (Population MCMC) algorithms, and can improve the efficiency of the standard MCMC, where only one point is accepted or rejected at each step.

7.2 Example

Here, a simple example of PMC in the context of parameter estimation in the Bayesian framework (the general problem in this work) is presented. Let us consider a simple model with 2 unknown parameters:

$$y = b_1(1 - e^{-b_2 x}) + \epsilon \tag{7.5}$$

where data is created with $\epsilon \sim N(0, 0.02^2)$ and $b = (1, 0.1)$. The PMC algorithm was run so that the mean and the covariance of the Gaussian importance distribution was updated at each iteration using the sampled points (in AM style). 500 points were created at each iteration and 10 iterations were made. The initial covariance was calculated from the data using the approximation from section 5.2. The Jacobian was calculated analytically. The scatter plot of the parameters is presented in figure 7.1.

The results seem to be somewhat similar and the maximum likelihood estimate for the point is close to the true values. The plot produced by MCMC is presented in figure 7.2. It shows similar behavior for the posterior distribution of the parameters. That is, the PMC algorithm is here successfully applied to a simple parameter estimation task. Thus, PMC is relatively simple to implement. Also, as mentioned, the adaptive schemes can be designed more loosely, since we do not have to deal with ergodicity conditions.

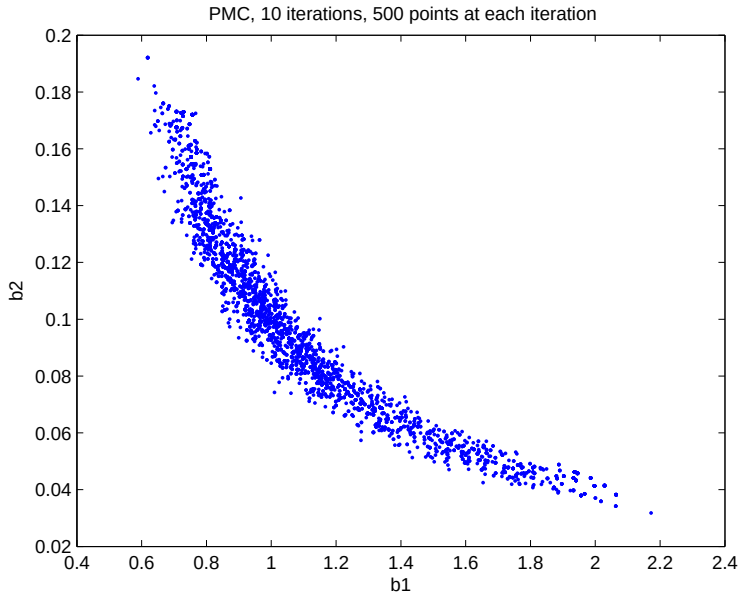

Figure 7.1: Population Monte Carlo with an adaptive importance distribution.

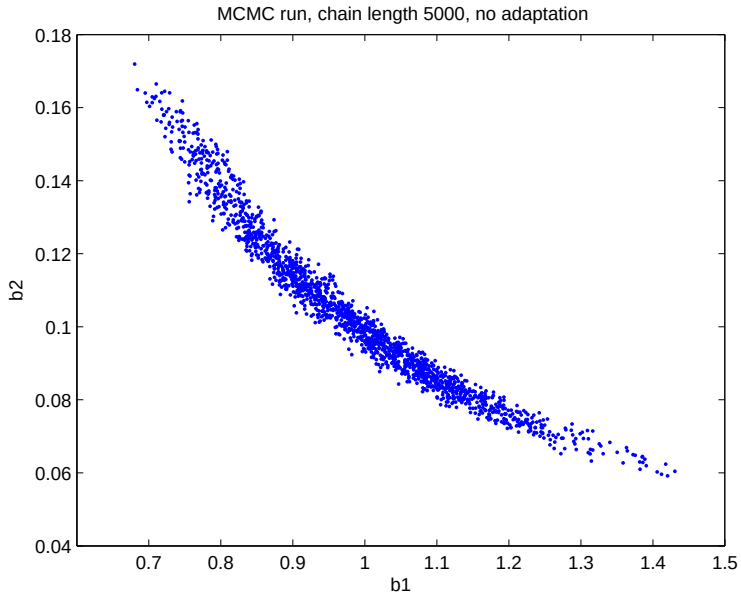

Figure 7.2: MCMC run for the example model

8 Developing MC Methodology

In this chapter, some aspects and ideas related to the future research among Monte Carlo methods in parameter estimation tasks are discussed. The purpose is merely to propose some ideas and directions for future development that came up when working with the methods. The ideas are related to sampling the error structure, forming new, effective proposal distributions and designing Population Monte Carlo schemes.

In MCMC methodology, a crucial question is the modeling of the error structure. Especially, when using a Gaussian error, the error variance can be sampled using conjugate priors, as briefly introduced in section 5.6. If the measurement errors are not independent and identically distributed ($\epsilon \sim N(0, \sigma^2 I)$), we should sample the *error covariance*, that can be done using a conjugate priors as well. This kind of sampling schemes are one area of potential development in MCMC methods.

The choise of the proposal distribution is essential when it comes to the efficiency and theoretical correctness of the algorithms. In the PMC approach, a good importance function seems to be an even more crucial issue. That is, one direction of development is to find new, effective proposal (importance) distributions. In particular, one could use, instead of one Gaussian distribution, a mixture of multiple Gaussians. A Gaussian mixture model (GMM) can be fitted to a data set, which allows the mixture proposal to adapt. At least intuitively a mixture proposal would be more efficient than a single Gaussian, at least in strongly "twisted" cases. A limiting factor in using a mixture model can be the additional computational burden. This can be helped if the previous classification information (GMM fit) can be used to construct the next fit in a recursive manner, as in updating the covariance in adaptive methods (chapter 6).

Another issue, worth looking into, might be different Principal Component Analysis tools for nonlinear cases. The information about principal components might be of great help when constructing the proposal distribution.

More effective proposal distribution could potentially be formed also using some information directly received from the model. For example, when approximating the covariance of the proposal using the inverse Hessian instead of the Jacobian approximation, we could get the inverse Hessian directly from a quasi-Newton style optimization algorithm such as BFGS.

Population Monte Carlo offers an interesting alternative to MCMC. And, as mentioned in chapter 7, it offers more freedom in designing adaptive importance distributions. The adaptive PMC algorithms might be an interesting field for future research. Also the PMC and MCMC can be coupled in many ways, for example MCMC can be used in diversifying the PMC sample after re-sampling. In addition, the "center points" for the importance distribution can be sought using MCMC - in this approach the only difference to MCMC methods would be multiple points created at each step with the re-sampling procedure.

When it comes to adaptive methods, one of the tuning parameters is the adaptation interval, which is normally kept quite large (100-200). The interval is kept large partly because in a bigger interval the algorithm most likely moves at least a little, which makes the adaptation (calculation of the covariance) possible. One idea would be not to keep the adaptation interval constant. We could do the adaptation based on the information about the movement of the algorithm. We could perform the adaptation, when a certain amount of new points are accepted. In this approach the adaptation interval would be bigger in the beginning and smaller in the end - and it would require less tuning from the user since the adaptation interval would not have to be defined explicitly.

9 Case Examples

This section includes the empirical part of the work. Here, a few models from the field of chemical reaction kinetics are presented and the distributions of the unknown parameters are formed with MCMC methods. Results are visualized with the methods presented in the theoretical part. The models are currently under research in Åbo Akademi (Turku, Finland) and in Lappeenranta University of Technology (Lappeenranta, Finland). First, a simple example model from chemical kinetics is presented. This model is used in some illustrations in the theoretical part of the work.

For solving the parameter estimation tasks in the case examples, two techniques are used. For MH, AM and DRAM solutions, the *MODEST* statistical software is used. The software and its MCMC tool are programmed using Fortran and mathematics libraries *LAPACK* and *BLAS*. On the other hand, for plotting purposes and for different modifications of the Metropolis and AM algorithms, the MATLAB software is used.

The emphasis of the theoretical part is on questions raised by the MCMC -type statistical analysis of the models, not on the actual chemistry behind the models and what the results mean from the chemical point of view. The latter part is left for the modelers themselves.

Most of the models are run with both Metropolis algorithm (section 4.2) and Adaptive Metropolis (section 6.1) in order to compare the methods and see if AM provides better results than the Metropolis algorithm. In two models, the DRAM algorithm is used.

9.1 Model Types and Data

The empirical part of this work concentrates on mechanistic mathematical models that describe chemical reactions. The general mechanistic model can be written as follows.

$$\mathbf{s} = f(\mathbf{x}, \theta, \mathbf{c}) \tag{9.1}$$

$$\mathbf{y} = g(\mathbf{s}) \tag{9.2}$$

where $\mathbf{s}$ is the *state* of the system with certain values for design variables $\mathbf{x}$, unknown parameters θ and constants $\mathbf{c}$. The function f represents the model. The observation function g denotes the relation between the states given by the model and the quantities observed (measured) by the modeler.

In so called *explicit algebraic* models the states can be explicitly calculated by substitution from the formula $\mathbf{s} = f(\mathbf{x}, \theta, \mathbf{c})$. This is the most simple case and no numerical solvers are needed to compute the states with different values for design variables.

The models describing chemical reactions are often ODE-systems, in which the concentrations of certain substances change in time. In ODE models the states are presented at each time point in the form

$$\frac{d\mathbf{s}}{dt} = f(\mathbf{s}, \mathbf{x}, \theta, \mathbf{c}) \tag{9.3}$$

In addition, some initial condition $\mathbf{s}(0) = \mathbf{s}_0$ is given. In order to obtain the state at each time point, the ODE has to be solved, which can be done analytically, or, in most cases, with a numerical solver. The models in this chapter are described as ODE systems. The solvers used in these models are *ODESSA* in the FORTRAN framework and *ode45* in MATLAB.

If the model is described as a system of differential equations that contain derivatives with respect to several variables, we end up to Partial Differential Equations (PDE). Normally PDE models include time and space variables. The PDEs can be transformed into ODE systems with discretization methods, for example. Often the computation time for function evaluations, and thus MCMC methods, rise.

The measurement data is often given as many *batches* (data sets). The model can include so called *local variables* that are constant within data sets but may differ between them, for example temperature. In addition, some *control variables*, that also change within the batches, may exist.

All in all, the model data is given in N_{set} batches. A batch k has $N_{obs}(k)$ observations and each observation j has $N_{resp}(j,k)$ measured response components. That is, we have measurements y_{ijk} where i is the index of the response component in observation j in batch k. If $y_{p_{ijk}}$ denotes the corresponding model prediction, the LSQ function to be minimized can be calculated as a sum over all batches, all observations and all response components as follows:

$$LSQ(\theta) = \sum_{k=1}^{N_{set}} \sum_{j=1}^{N_{obs}(k)} \sum_{i=1}^{N_{resp}(j,k)} (y_{ijk} - y_{p_{ijk}})^2. \tag{9.4}$$

In table 2 below, the models dealt with in this chapter are summarized. For each model, the model type, the MCMC algorithm used, the dimension and the number of observed

responses, are listed.

Table 2: Example model types, algorithms used and number of unknown parameters and observed responses.

Name	Section	Model	Algorithm	Dim	Resp
Example Model	9.2	ODE (2 components)	MH/AM	4	2
Phloroglucinol	9.3	ODE (3 components)	AM	7	3
Propionic Acid	9.4	ODE (6 components)	AM	7	4
Esterification	9.5	ODE (4 components)	AM	5	1
Sitosterol	9.6	ODE (14 components)	DRAM	22	11

Thus, the number of unknown parameters in the models range from 4 to 22. In most of the cases, the AM algorithm gave good results, and the parameters were identified well. In one case (sitosterol), due to the complex nature of the posterior distribution, the acceptance ratio was very low with MH and AM, and the DRAM algorithm was used to get the sampler moving. In this case there were also problems in the identifiability of some parameters.

In the "difficult" cases with many unknown parameters and ODE components, the computational time was significantly larger than that of the simpler models; the computation of a sufficiently long MCMC chain could take about one day when using a standard workstation. In a very complicated model with a very large number of unknown parameters, a more powerful computing environment would be required.

9.2 Simple Example Model

This model is used for illustration purposes in previous chapters. The model describes a simple one-way chemical reaction, where A is transformed into B with some speed described by rate parameter k_1, and B is disappearing with rate k_2. That is, the reaction can be written as

$$A \rightarrow^{k_1} B \rightarrow^{k_2}$$

which means that the model can be defined as an ODE system as follows

$$\begin{aligned} \frac{dA}{dt} &= -k_1 A \\ \frac{dB}{dt} &= k_1 A - k_2 B \end{aligned}$$

where k_1 and k_2 are reaction rates. The temperature dependency of the reaction rate is expressed in chemical kinetics using the Arrhenius' law, in which

$$k_1 = k_1^{mean} e^{-E_1 z} \tag{9.5}$$

$$k_2 = k_2^{mean} e^{-E_2 z} \tag{9.6}$$

$$z = 1/R(1/T - 1/T_{mean}). \tag{9.7}$$

The frequency factors k_i^{mean} denote the "average" reaction rate is some "average" temperature T_{mean}. R is the general gas constant and E_1 and E_2 denote the activation energies for the reaction. The estimated parameters here are $\theta = (k_1^{mean}, E_1, k_1^{mean}, E_2)$. T denotes the temperature in which the measurements are made (a local variable).

To illustrate the general notation given in chapter 9.1, the states and the observation function can be defined as

$$\begin{aligned} s(1) &= A \\ s(2) &= B \\ g(s) &= s. \end{aligned}$$

That is, we observe both components of the ODE system. If only the second component was observed, we would have $g(s) = s(2)$. In this model $\theta = (k_1^{mean}, E_1, k_1^{mean}, E_2)$ and $x = (T, A(0), B(0))$. The constants are $c = (R, T_{mean})$.

The MCMC results for the example model were presented in section 5.5. In the follow-

ing, an additional MCMC example of optimizing the temperature profile for the example modes is presented.

9.2.1 Optimizing the Temperature Profile

Let us consider the temperature T in the example model to be a control variable, which means that it is allowed to vary between different time points. Then, a useful question in practice would be to think how we should control the temperature so that, for example, the output of the intermediate component B is maximized.

Let $T = (T_1, ..., T_n)$ denote the temperature profile, that we can control. Here T_i denotes the temperature in time point t_i. Now we optimize the temperature profile so that the output of B is maximized. Thus, for optimal temperature profile T^* we can write

$$T^* = \max_T B(t_n).$$

The resulting optimal temperature profile is presented if figure 9.1. From the results we can conclude that, in order to get a large final value for B, we must keep the temperature low in the beginning of the reaction and then quickly increase it in the end. The simplified temperature profile, that can be implemented during the experiment, is presented in figure 9.1.

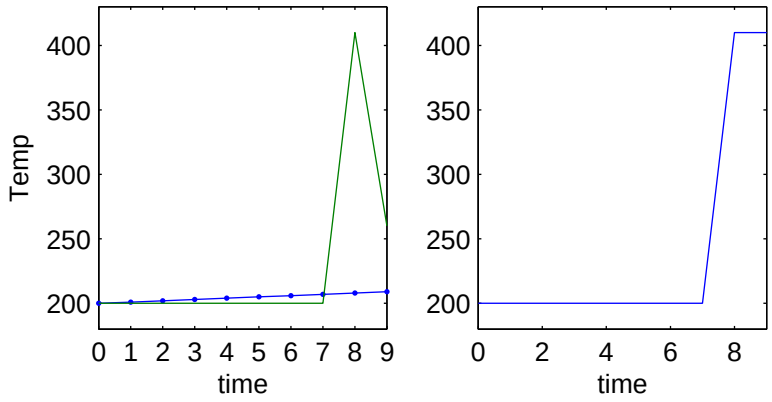

Figure 9.1: The optimized temperature profile (with an initial guess) and the corresponding simplified profile.

Now we can simulate the responses A and B with the optimized temperature profile, with the different possible parameters given by the MCMC run, to see how responses behave with the optimized temperature profile. The results for both response components are

presented in figure 9.2. The results show that the uncertainty in the model is dramatically increased in comparison to the original setting (figure 5.10). That is, when we move away from the original temperature profiles, where the measurements were made, the predictability of the model deteriorates: Thus, the model is not able to accurately explain the behaviour of the response components when the conditions of the experiment are changed: the uncertainty in the activation energy estimates produce a large uncertainty in the new situation. The MCMC methodology is easily able to reveal this kind of prediction error.

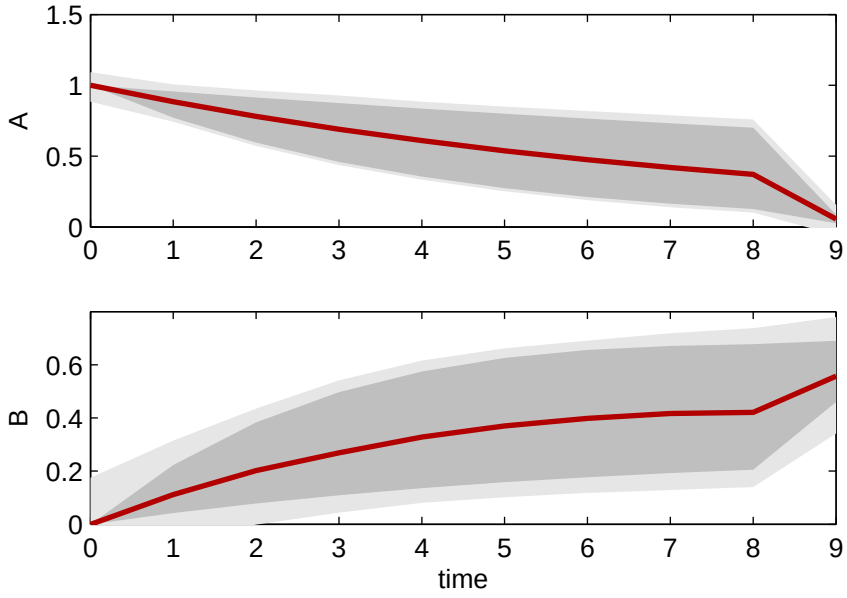

Figure 9.2: The predictive distributions for the responses with the optimized temperature profile.

9.3 Creation of Phloroglucinol

The creation of phloroglucinol is a process studied in the department of Chemical Engineering at Lappeenranta University of Technology. The model is formulated as an ODE system with 3 response components, all of which are observed. The number of unknown parameters is 7. The measurement are done in 5 experiments in different temperatures.

From the MCMC perspective the parameters seem to be rather well identified. The AM algorithm seems to provide slightly better mixing than the standard MH algorithm. The predictive distributions show that the model fits well to the data.

9.3.1 Model Description

The kinetic model for the overall reaction is formulated as an ODE system as follows.

$$\begin{aligned} \frac{dA}{dt} &= -M_r^{n_1} R_1 - M_r^{n_2} R_2 \\ \frac{dB}{dt} &= -2M_r^{n_1} R_1 \\ \frac{dC}{dt} &= M_r R_2 \end{aligned}$$

where

$$\begin{aligned} R_1 &= k_1 AB \\ R_2 &= k_2 A^{n_3}. \end{aligned}$$

The rate constants k_1 and k_2 and z are defined as in the example model in the previous chapter (Arrhenius' law). That is, the parameters to be estimated here are $\theta = (k_1^{mean}, E_1, k_2^{mean}, E_2, n_1, n_2, n_3)$. The states are defined to be $\mathbf{s} = [A, B, C]$ and the observation function $g(\mathbf{s}) = \mathbf{s}$.

9.3.2 MCMC Results

Table 3 below shows the LSQ estimate, empirical median and confidence interval limits ($\alpha = 0.05$) for the model. From the estimates one can see that, for example, E_1 and E_2 are well identified, whereas the values sampled for n_2 and n_3 get a wider range of values.

Table 3: LSQ estimate and empirical confidence limits for the model.

	LSQ	0.5	$\alpha/2$	$1-\alpha/2$
k_1^{mean}	0.274E-03	0.295E-03	0.179E-03	0.4469E-03
E_1	0.273	0.274	0.264	0.285
k_2^{mean}	0.0098	0.010	0.0045	0.022
E_2	0.213	0.221	0.201	0.244
n_1	2.11	2.08	1.95	2.23
n_2	0.624	0.736	0.330	1.221
n_3	0.598	0.756	0.471	1.085

The pairwise scatter plots are presented in figure 9.3. The plot shows that most of the parameters behave nicely. The scatter plot reveals that there is correlation between some parameters, the strongest correlation being between parameter pairs (6,7) and (3,6). The correlation coefficients (see Appendix 1 for definition) calculated from the chain are about 0.8 for both of the pairs.

When we look at the marginal plots in figure 9.5, the correlations are also clearly shown. The figures on the right, that represents a chain created with adaptive metropolis, seems to be mixing more efficiently than the plain Metropolis MCMC presented on the left side.

Predictive distribution curves for one of the data sets (altogether 5) are presented in figure 9.4 using the error variance sampling scheme presented in section 5.6. The parameters in the prior distribution for the error variance were defined to be $n_0 = 1$ and $S_0^2 = MSE$. If a prediction is made from the response curves, it should roughly hold with the accuracy given by the lighter gray area in the paint brush curve.

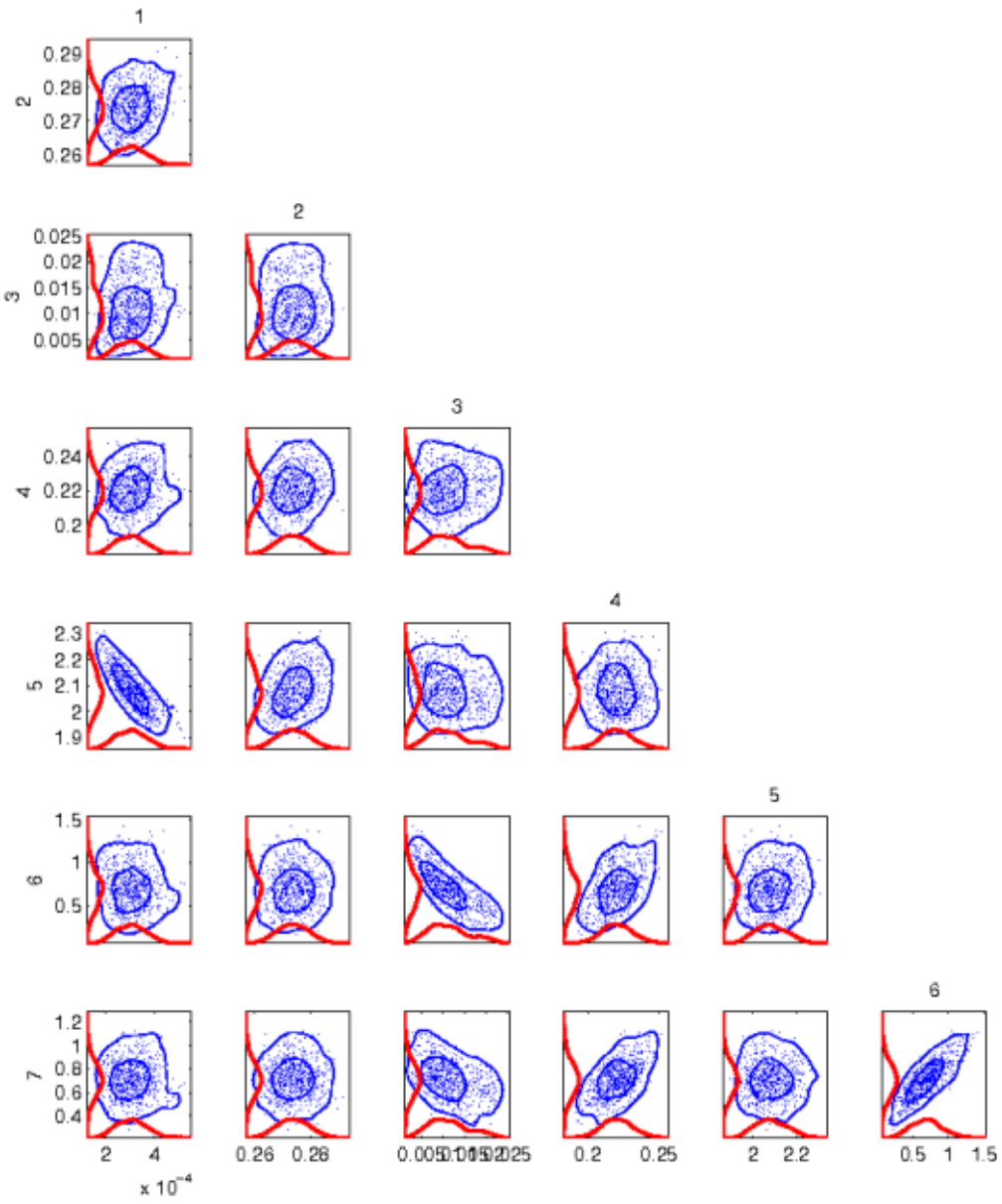

Figure 9.3: Pairwise scatter plots from the chain created with Metropolis algorithm. The burn-in time was 1000 and chain length 10000. One-dimensional confidence intervals and confidence regions (approximately 95%) are created with the kernel density method.

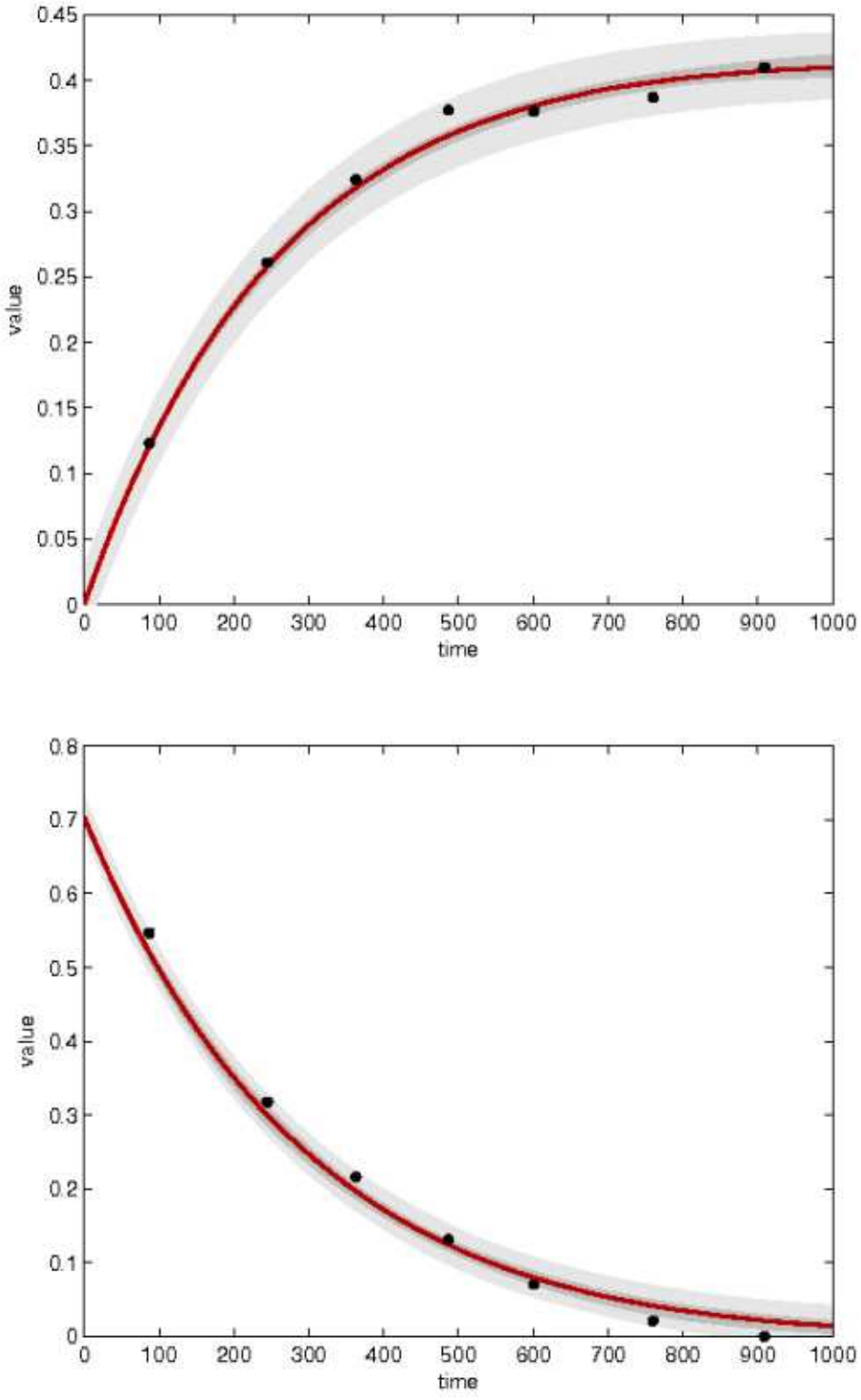

Figure 9.4: Distributions for model fits. The chain was created using AM with adaptation interval 100 and burn-in period of length 1000.

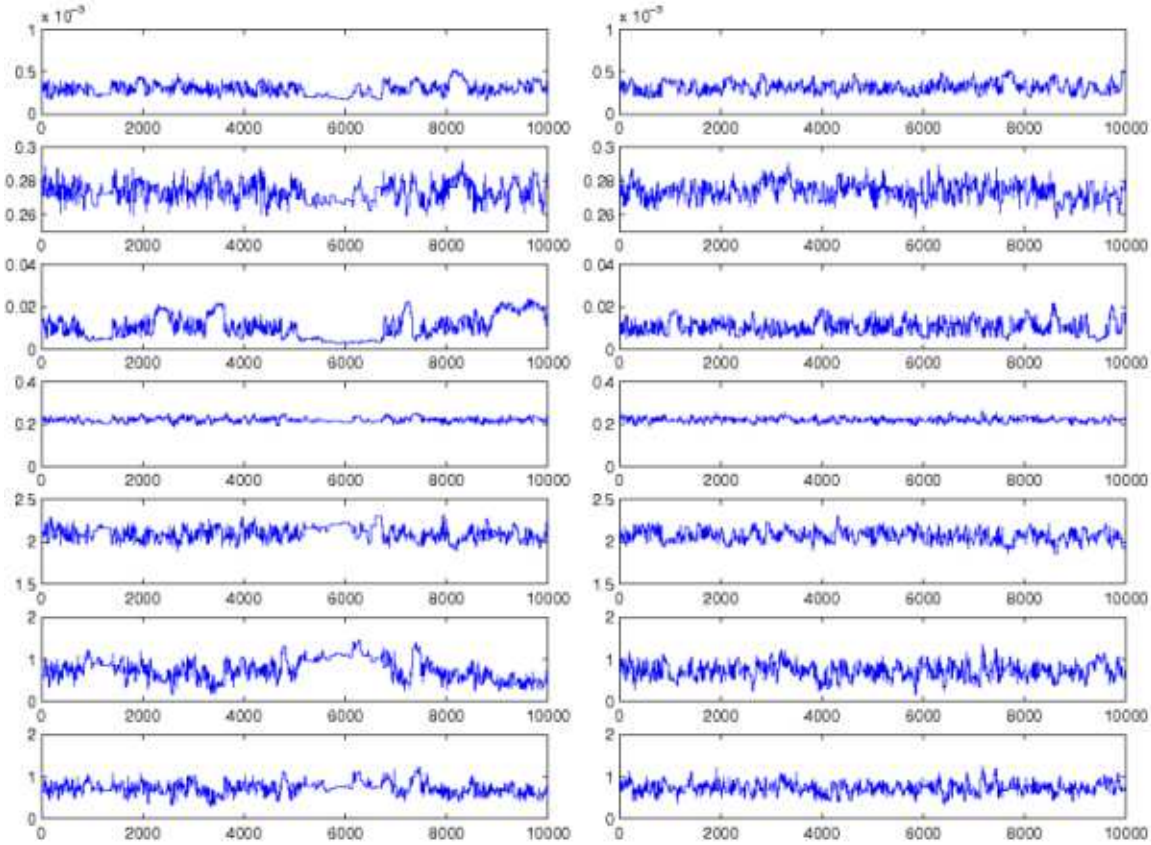

Figure 9.5: Marginal Paths for the 7 unknown parameters. The left picture is created with plain Metropolis MCMC, the right one with AM, using adaptation interval 100 and regarding the initial covariance matrix as if it was sampled from 1000 points.

9.4 Esterification of Neonpentyl Glycol with Propionic Acid

The model is studied at Lappeenranta University of Technology, at the Department of Chemical Engineering. It is described as a 6-component ODE-system, 4 of which are observed. The reaction kinetics are studied in a batch reactor at different temperatures. The measurement are given in 12 batches and the number of unknown parameters is 7. For a thorough explanation of the chemistry behind the ODE is given in [40] (to be published).

From the MCMC perspective the model parameters seem to behave well, and the standard algorithms, MH and AM, are well suitable for the evaluation of the posterior distribution of the parameters.

9.4.1 Model Description

The model can be defined as a 6-component ODE system as follows.

$$\begin{aligned}
\frac{dA}{dt} &= -k_1AB + \frac{k_3C^2}{K_{eq}} - k_3AD - \frac{A}{M}W \\
\frac{dB}{dt} &= -k_1AB - k_2BC - \frac{B}{M}W \\
\frac{dC}{dt} &= k_1AB - k_2BC + 2k_3AD - \frac{2k_3C^2}{K_{eq}} - \frac{C}{M}W \\
\frac{dD}{dt} &= k_2BC + \frac{k_3C^2}{K_{eq}} - \frac{D}{M}W \\
\frac{dE}{dt} &= (k_1AB + k_2BC)M \\
\frac{dM}{dt} &= -W
\end{aligned}$$

where

$$W = M_w(k_1AB + k_2BC)M.$$

Here M_w is a given constant and the reaction rates k_1, k_2 and k_3 are derived from the Arrhenius law, and thus the unknown parameters are $\theta = (k_1^{mean}, E_1, k_2^{mean}, E_2, k_3^{mean}, E_3, K_{eq})$.

In the "state notation" we can write all the calculated states as follows:

$$\begin{aligned}
[s(1)...s(6)] &= [A, B, C, D, E, M] \\
s(7) &= (A(0) - MA)/A(0) \\
s(8) &= (B(0) - MB)/B(0) \\
s(9) &= (MC - C(0))/(A(0) - MA) \\
s(10) &= (MC - C(0))/(B(0) - MB) \\
s(11) &= (A + B + C + D)M + E.
\end{aligned}$$

Components 7 and 8 represent the *conversion* for A and B and components 9 and 10 the *selectivity* for A and B. The 11th state records the total mass of the reaction components. We observe the first 4 components of the ODE system. That is, the observation function is written as

$$g(\mathbf{s}) = [s(1), s(2), s(3), s(4)]\,.$$

9.4.2 MCMC Results

The pairwise scatter plots presented in figure 9.6 show that the parameters are well identified. In this kind of case, where the posterior distribution behaves in a regular way, the AM algorithm does not provide significant improvement compared to the MH algorithm with MSE chosen as the error variance.

The predictive distributions for the states in three data sets are given in figures 9.7 and 9.8. We see that responses A and B fit well to the data, but there is systematic error in the model with respect to responses C and D, that overestimates the measurement error with respect to these responses. The question about how to include the error in the model itself into the statistical analysis, in addition to the measurement error, is subject to ongoing research and not addressed here.

The same kind of temperature profile optimization as for the example model in chapter 9.2 can be carried through for this model. For implementation and results, refer to [40].

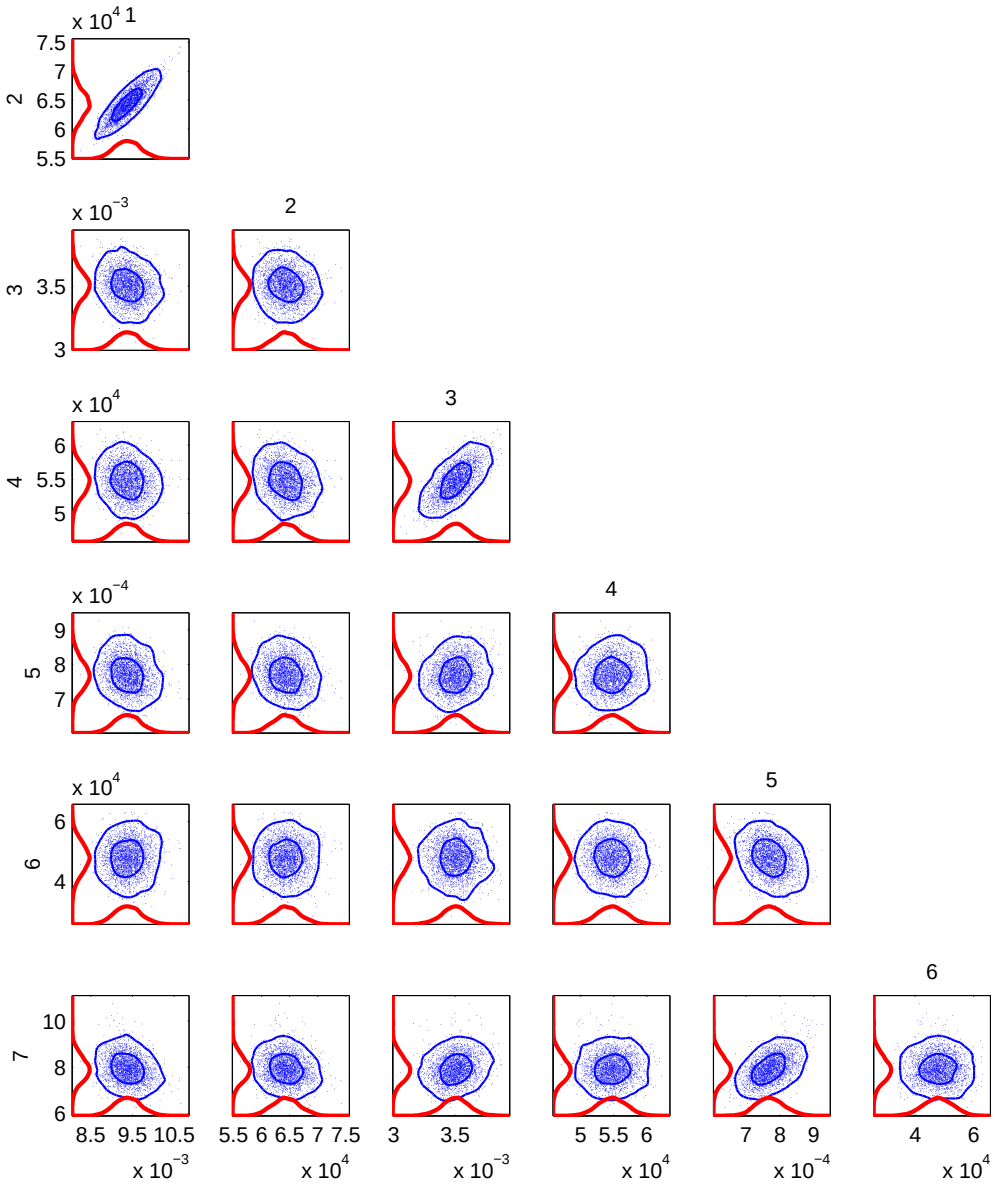

Figure 9.6: Pairwise scatter plots for the model parameters. The chain was produced with AM, adaptation interval being 100.

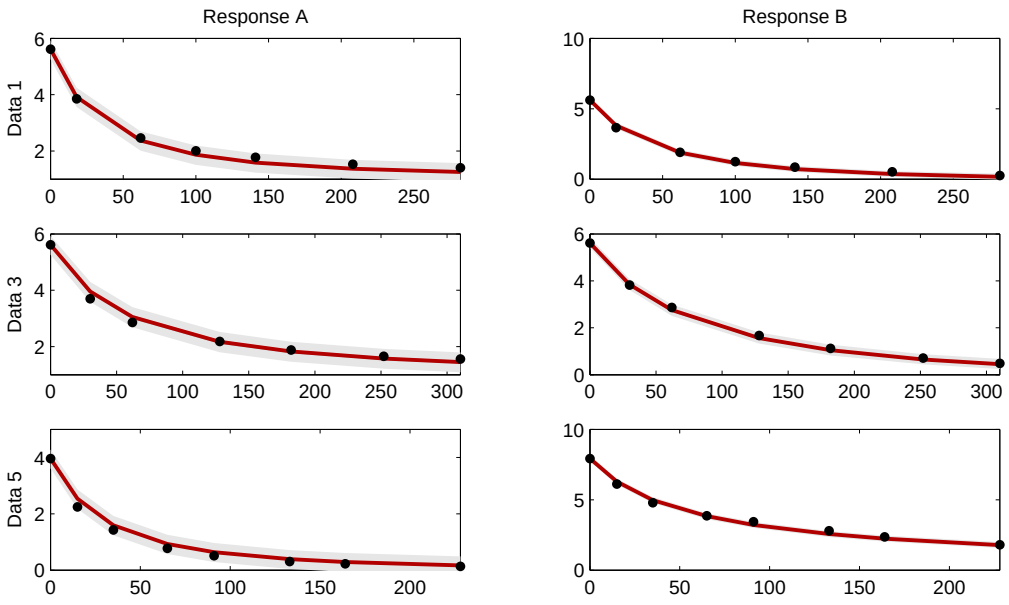

Figure 9.7: Predictive distributions with good fits.

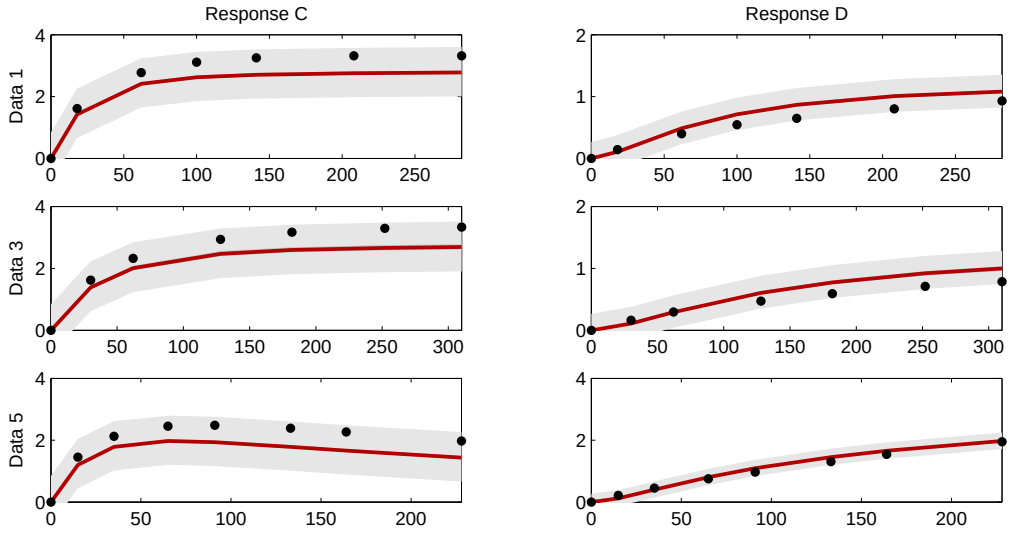

Figure 9.8: Predictive distributions with systematic error in the model.

9.5 Esterification Processes

The model discussed here has been studied in Åbo Akademi, Turku, Finland. The description of the chemistry behind the model can be found from [41]. The model is an ODE system with 4 components, one of which is observed. The number of unknown parameters is 5 and the measurements are done in 9 batches.

From the MCMC perspective the model parameters seem to behave in a satisfactory way. The AM algorithm seems to result in slightly better mixing than MH. The predictive distribution reveals systematic error in the observed response.

9.5.1 Model Description

The model is described as an ODE system below.

$$\begin{aligned}
\frac{dc_A}{dt} &= -r \\
\frac{dc_B}{dt} &= -r \\
\frac{dc_C}{dt} &= r \\
\frac{dc_D}{dt} &= r
\end{aligned}$$

where

$$\begin{aligned}
r &= \frac{c_A c_B - c_C c_D / K}{(\alpha c_A + \beta c_D) c_h \frac{V_0}{V}} \\
\alpha &= \alpha_0 e^{E_1/8.314z} \\
\beta &= \beta_0 e^{E_2/8.314z} \\
z &= 1/T - 1/T_{mean}.
\end{aligned}$$

The parameters to be estimated are $\theta = (K, \alpha_0, \beta_0, E_1, E_2)$. The initial volume V_0 is a local variable and volume V is a control variable, that may change in every measurement within batches. Here $\mathbf{s} = [A, B, C, D]$ and we observe only the first component of the system: $g(\mathbf{s}) = s(1)$.

9.5.2 MCMC Results

In this model chain length of 20000 and burn-in time of 5000 were used. Different adaptation intervals were tried and also different burn-in schemes - both fixed and adaptive proposals (scaling and greedy adaptation) were used.

Table 4 below shows the LSQ estimate, empirical median and confidence interval limits for the model parameters, calculated from the MCMC chain.

Table 4: LSQ estimate and empirical confidence limits for the parameters.

	LSQ	0.5	$\alpha/2$	$1-\alpha/2$
K	2.92	3.02	2.34	3.92
α_0	3.29	3.20	1.93	4.47
β_0	23.0	23.6	18.2	29.5
E_1	66300	66000	23500	112000
E_2	41500	42800	13500	71200

Two marginal path plots are presented in figure 9.9. One can again see that with the adaptive method the mixing of the chain seems to be significantly better, especially with respect to the first parameter (K). The mixing is significantly improved merely by employing scaling and / or greedy adaptation during the burn-in phase. If adaptation is used also after the burn-in, the mixing is slightly improved. That is, here the greedy adaptation procedure turns out to be especially useful.

The pairwise scatter plot produced by AM (with adaptation interval 100 and greedy adaptation during the burn-in) is presented in figure 9.10. All parameters seem to have one clear optimum. The strongest correlation exists between parameters (3,2).

The predictive distributions for the first response variable is presented in figure 9.11 for 4 data sets. The plot shows that the model seems to fit the data accurately.

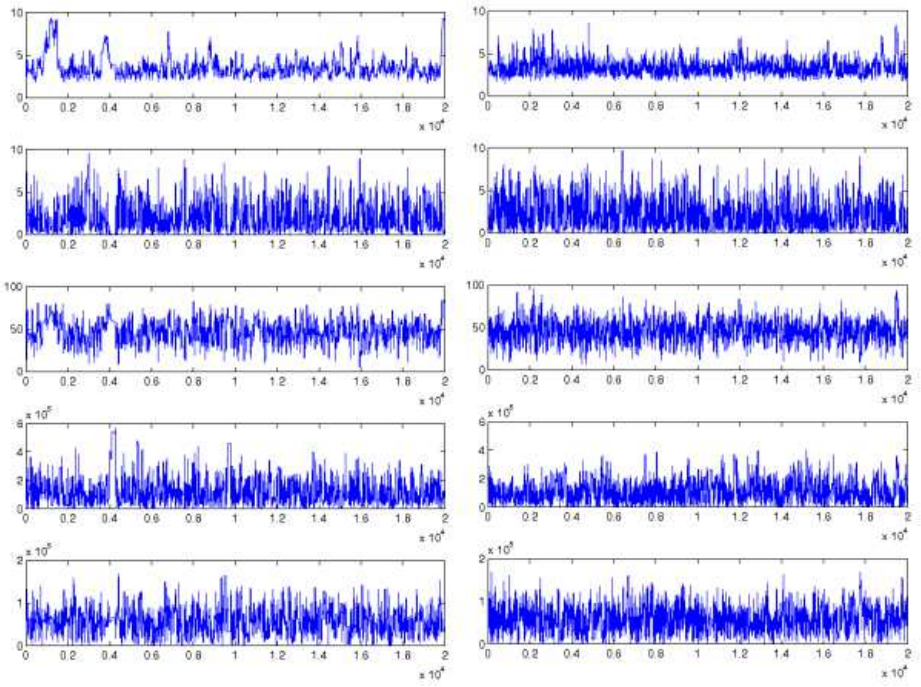

Figure 9.9: Marginal Paths for the 5 unknown parameters. The left picture is created with plain Metropolis MCMC, the right one with AM, using adaptation interval 100 and greedy adaptation during the burn-in phase.

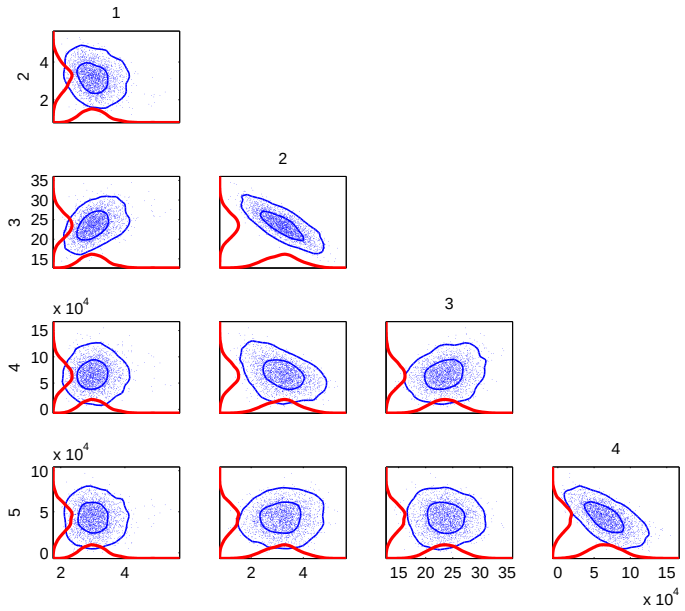

Figure 9.10: Pairwise scatter plots from the chain created with the AM algorithm. The burn-in time was 5000 and chain length 20000. Adaptation interval was 100.

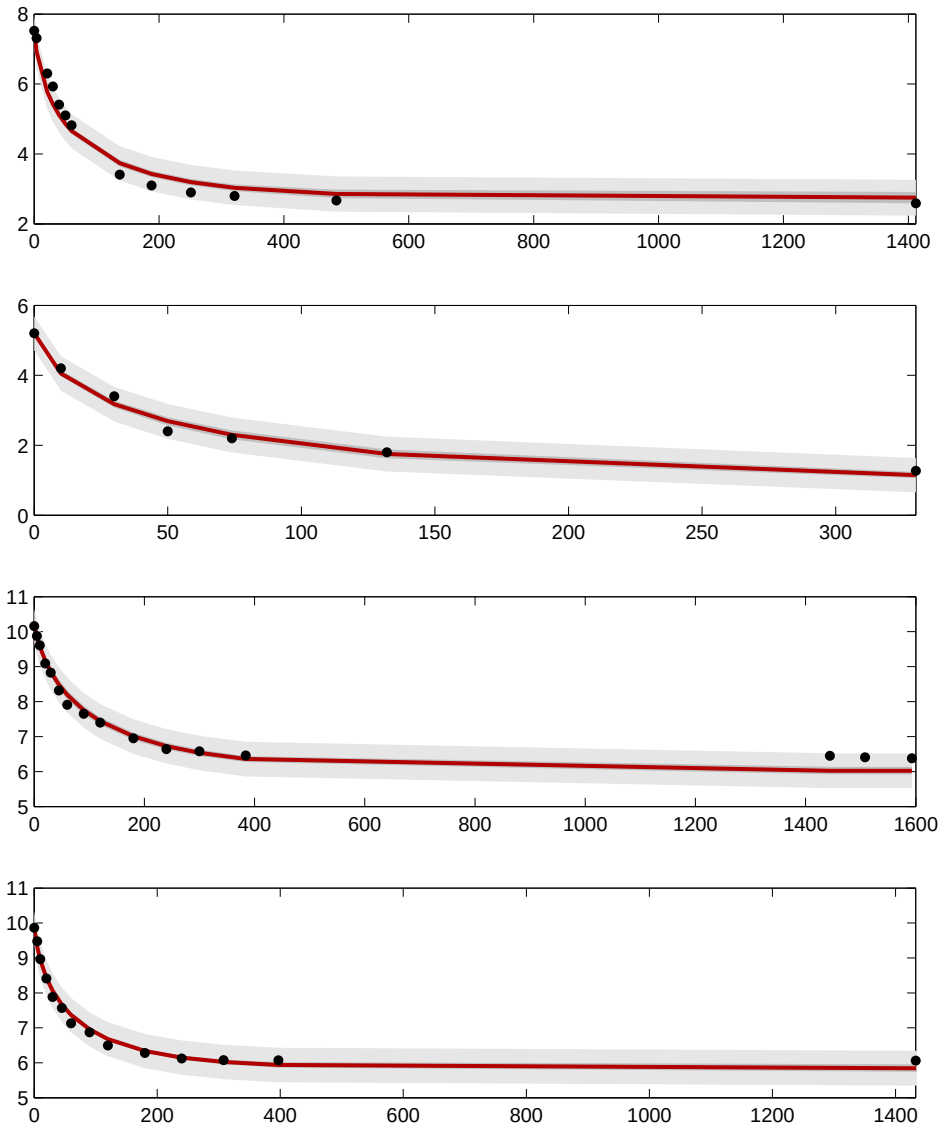

Figure 9.11: Predictive distribution curves for 4 data sets.

9.6 Sitosterol Hydrogenation Process

The model, studied in the Process Chemistry centre of Åbo Akademi (Turku, Finland), is defined as an ODE system of 14 components, 11 of which are observed. The number of unknown parameters is 22. The mathematical description of the model is omitted here due to its complexity - for the description refer to [42].

From the MCMC point of view this model is interesting, because the dimension and the number of observed response components are higher than those of the other cases. The basic MCMC algorithms, MH and AM, seem to have problems in getting themselves moving: the acceptance ratio is very low. The DRAM algorithm provides significant help in this case. The distribution of the parameters reveals strong correlation between two parameters, which could justify model reduction. Also, a few parameters seem to be badly identified and some parameters converge close to the optimization limits. The predictive distributions show that some of the responses are able to predict the measurements well and some predictive distributions include more noise.

9.6.1 MCMC Results

Some of the unknown parameters in the model are well identified (see figure 9.12). In some cases there seem to be clear identifiability problems, that can be seen from figures 9.13 and 9.14.

One parameter pair is very strongly correlated (figure 9.15) and it seems that the ratio between the parameters is well identified, but the parameters themselves are not. Model refinement and reparametrization could help in this issue. In addition, at least one parameter gets very close to zero and the parameter could be assigned a fixed value (zero) and removed from the optimization target.

The predictive distributions for the components in some data sets are presented in figures 9.16 and 9.17.

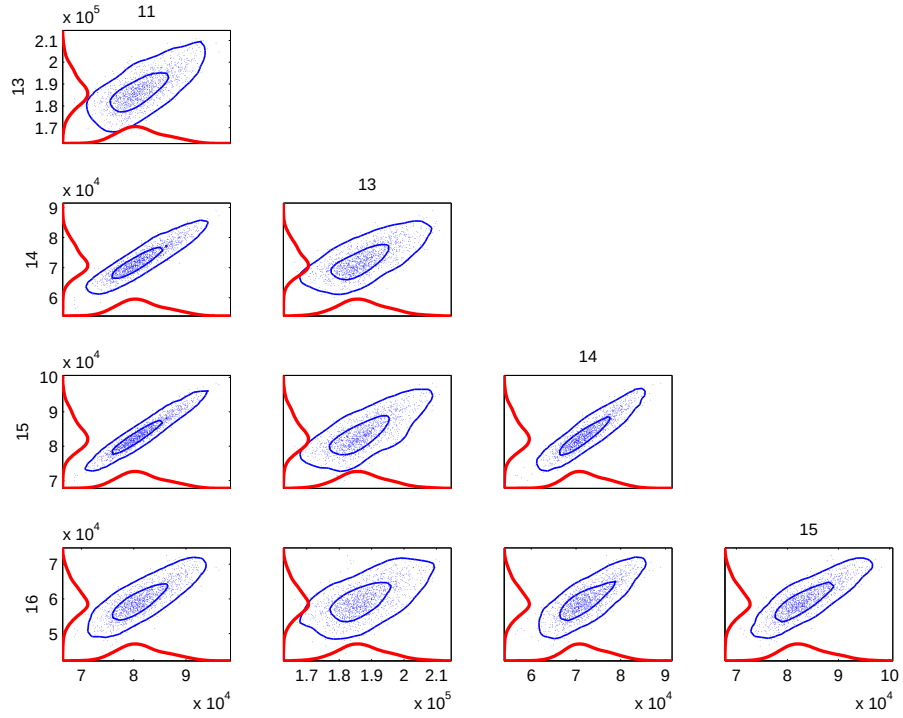

Figure 9.12: The well behaving model parameters.

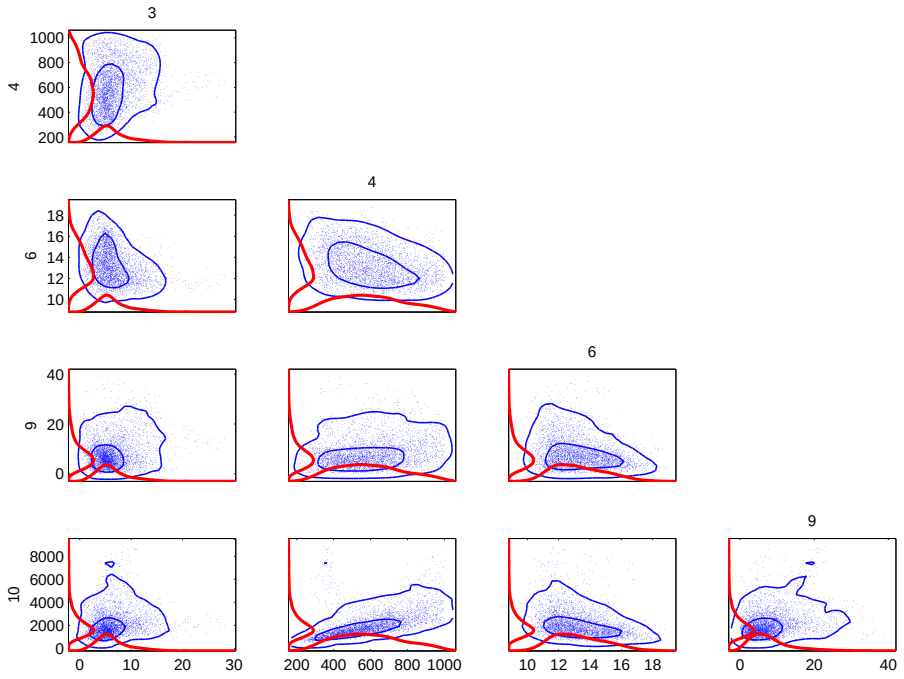

Figure 9.13: Problems, for example, related to the identifiability of parameter 4.

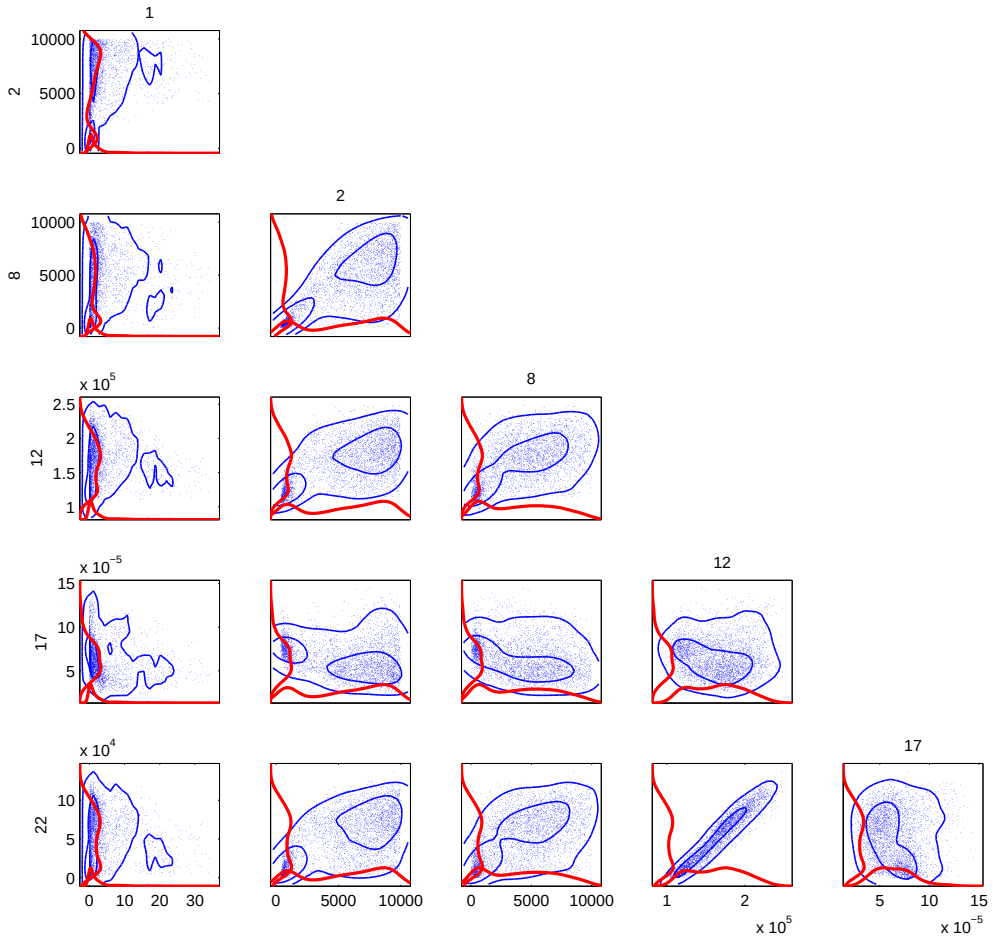

Figure 9.14: Problems in the identifiability of parameters. In addition, the first parameter seems to get values close to zero. Strong correlation between parameters 12 and 22.

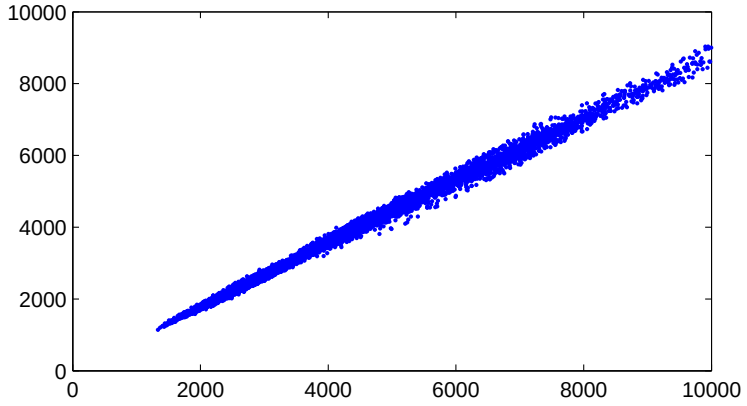

Figure 9.15: Strongly correlated parameters. Here reparametrization could be done so that the ratio of the parameters would replace the other parameter in the list of unknown parameters.

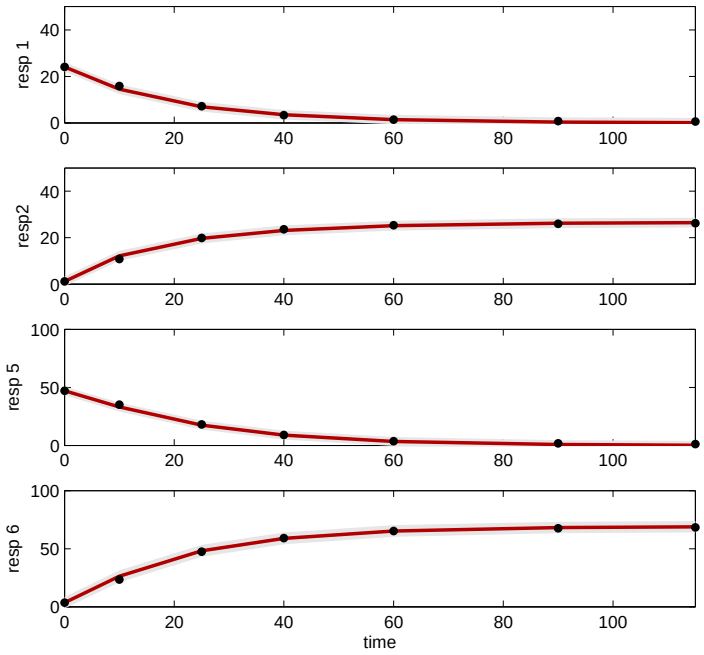

Figure 9.16: Predictive distributions for response components 1,2,5 and 6. The model prediction fits the measurements accurately.

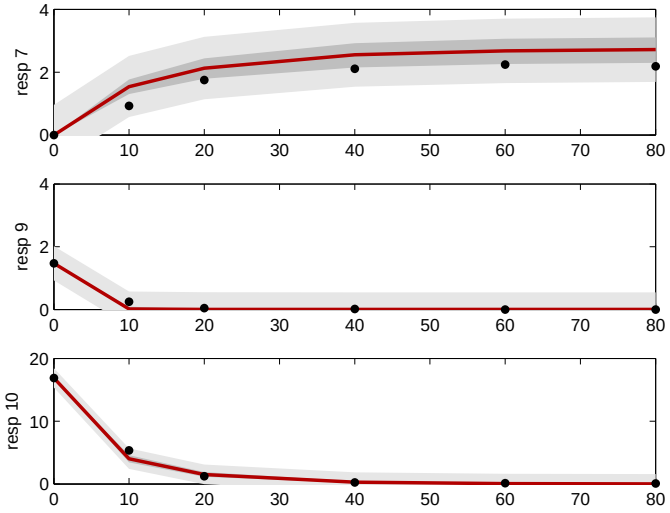

Figure 9.17: Predictive distributions for response components 7,9 and 10. Some noise occurs in the model prediction.

10 Conclusions

Due to increased computational power the MCMC methods have developed into a feasible tool for investigating the posterior distribution of unknown parameters in mathematical models. The methods are straightforward to implement and thus employable also in real life modeling tasks, as seen in the case examples of this work.

The numerical methods for sampling from the posterior distribution provide many kinds of new, valuable information to the modeler, including correlations between parameters, uncertainties related to both parameter estimates and prediction curves and identifiability of the parameters. When dealing with regular MCMC methods based on the Metropolis and Metropolis-Hastings algorithms, problems may however rise, if the posterior behaves in a certain way, for example if the posterior is strongly "twisted". The proposal distribution has to be correctly chosen in order to make the algorithms work efficiently. In this case, the family of adaptive methods provide significant help - the proposal distribution adapts to the shape of the posterior and the methods become less dependent on the initial proposal distribution. The effect of the adaptive methods can be seen in better mixing of the produced MCMC chain.

The adaptive methods seem to have a significant effect in the efficiency of the algorithm, especially in correlated cases. High-dimensional problems are often efficiently tackled with single component versions of the algorithms. If the initial point and the identifiability of the parameters is poor, the Delayed Rejection approach often provides better results.

There are many different aspects that are to be considered when building applications and code that implement some MCMC -type method. One of the main questions is convergence. It can be diagnosed with many methods, of which the most useful seem to be based visual examination of the output of the algorithm and discarding the initial "burn-in" period from the output. In addition, one has to be able to provide efficient starting points and initial proposal distributions through different estimates (LSQ) and approximations (linearization). Also the interpretation of results and visualizations is an important issue when making inference about the posterior distribution. The results may be visualized using marginal distributions with confidence intervals and confidence regions. In addition, the effect of the uncertainty in parameters to the prediction (response) curves can be visualized by building predictive distributions.

In adaptive methods, the main implementation issue is the adaptation interval - how often

should the adaptation be done? Normally the adaptation interval is kept relatively large in order to make the algorithm move in the beginning and thus assuring that there are enough diverse points to make the adaptation. Some special tricks can be employed to make the algorithm move, including greedy adaptation procedure. The DRAM algorithm seems to be helpful as well, when there are problems in getting the sampler moving. Also the role of the initial proposal distribution has to be considered - it is possible to discard it when the first adaptation occurs or assign a weight to it that represents the trust towards the initial proposal distribution.

In order to keep the theory and implementations simple, the measurement error is in this work assumed to be distributed in a Gaussian way and the errors in different measurement are assumed to be independent and identically distributed. The error can also be modeled in Bayesian terms so that the error variance is sampled using conjugate priors. If the assumption is that the errors are not independent and there can be correlation in the errors between the measurements, we end up in error covariance approximations and sampling schemes.

Population Monte Carlo methods, based on performing importance sampling iteratively, seem to provide a promising alternative to standard MCMC methods. The PMC methods also provide more freedom when designing adaptation schemes. This approach is subject to ongoing research.

References

[1] Kaipio, J. Somersalo, E. 2005. *Statistical and Computational Inverse Problems*, Applied Mathematical Sciences (Vol. 160). New York, USA: Springer Science+Business Media. 344 p. Chapter 3. ISBN 0-387-22073-9

[2] Tamminen, J. 1999. *MCMC Methods for Inverse Problems*. Licenciate Thesis. **Geophysical Publications** (Vol. 48). Helsinki, Finland: Finnish Meteorological Institute. 82 p.

[3] Gelman, A., Carlin, J., Stern, H. Rubin D. 1996. *Bayesian Data Analysis, Second Edition*. London, Great Britain: Chapman Hall. 696 p. ISBN: 0-158-48838-8

[4] Neal, R. 1993. *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Technical Report (CRG-TR-93-1). Toronto, USA: University of Toronto, Department of Computer Science. 144 p. Chapters 1-4. Available as PDF at: http://www.cs.toronto.edu/řadford/ftp/review.pdf.

[5] Nurmi, P. 2004. *Introduction to Bayesian Data Analysis*. The Seminar of Computational Data-analysis. Helsinki, Finland: University of Helsinki, Department of Computer Science. Available as PDF at: http://www.cs.helsinki.fi/u/salmenki/lda-seminaari04/bayesian.pdf.

[6] Robert, P. Casella, G. 2004. *Monte Carlo Statistical Methods*. New York, USA: Springer-Verlag. 645 p. Chapters 1-3,6-7,14. ISBN 0-387-21239-6

[7] Karlsson, R. 2002. *Simulation Based Methods for Target Tracking*. **Linköping Studies in Science and Technology** (Thesis No. 930). Linköping, Sweden: Linköping University, Department of Electrical Engineering, Division of Automatic Control and Communication Systems. Chapters 2-3.

[8] Råde, L. Westergren, B. 2001. *Beta, Mathematics Handbook*. Lund, Sweden: Studentlitteratur. 546 p. ISBN: 91-44-00839-2

[9] Gilks, W., Richardson, S. Spiegelhalter D. 1996. *Markov Chain Monte Carlo in Practice*. New York, USA: Springer Science+Business Media. 512 p. Chapters 1,3-4,19,22. ISBN 0-412-05551-1

[10] Mackay, D. 1996. *Introduction to Monte Carlo Methods*. In collection *Learning in Graphical Models*. Netherlands: Kluwer Academic Publishers. pp. 175-204. Available as PS at: http://www.cs.toronto.edu/m̃ackay/erice.ps.gz.

[11] Grinstead, C. Snell, J. 1997. *Introduction to Probability, Second Revised Edition*. USA, American Mathematical Society. Available as PDF at: http://www.dartmouth.edu/čhance/teaching$_a ids/books_a rticles/probability_b ook/pdf.html.ISBN$: $0 - 8218 - 0749 - 8$

[12] Schon, T. 2003. *On Computational Methods for Nonlinear Estimation.* **Linköping Studies in Science and Technology** (Thesis No. 1047). Linköping, Sweden: Linköping University, Department of Electrical Engineering, Division of Automatic Control and Communication Systems. Chapter 4.

[13] Haario, H. 2006. *Statistical Methods for Inverse Problems*. Lecture Material. Lappeenranta, Finland: Lappeenranta University of Technology.

[14] Doucet, A. 2000. *On Sequential Monte Carlo Sampling Methods for Bayesian Filtering*. Submitted for Publication. Available as Technical Report CUED/F-INFENG/TR. Cambridge University, Department of Engineering. Available at: http://citeseer.ist.psu.edu/article/doucet00sequential.html.

[15] Pursiainen, S. 2003. *Numerical Methods in Statistical EIT*. Masters Thesis. Espoo, Finland: Helsinki University of Technology. Chapter 3.

[16] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. Teller, E. 1953. *Equation of State Calculations by Fast Computing Machines*. **The Journal of Chemical Physics** (21/06). pp. 1087-1092

[17] Tierney, L. 1994. *Markov chains for exploring posterior distributions.* **The Annals of Statistics** (22). pp. 1701-1762

[18] Haario, H., Saksman, E., Tamminen, J. 2006. *Componentwise adaptation for high dimensional MCMC.* **Computational Statistics** (Vol 20, no 2). pp. 265-274.

[19] Rao, S. 1996. *Engineering Optimization: Theory and Practice, 3rd Edition*. New York, USA: John Wiley and Sons. 920 p. ISBN: 0-471-55034-5

[20] Brooks, S. Roberts, G. 1998. *Assessing Convergence of Markov Chain Monte Carlo Algorithms*. **Statistics and Computing** (Vol. 8). pp. 319-335. Available at: http://citeseer.ist.psu.edu/andrieu03introduction.html.

[21] Venables, W. Ripley, B. 1997. *Modern Applied Statistics with S-PLUS, Second Edition.* New York, USA: Springer. 548 p. ISBN 0-387-98214-0

[22] Hartigan, J. Wong M. 1979. *K-Means Clustering Algorithm.* Applied Statistics (Vol 28, no. 1). pp. 100-108.

[23] Figueiredo, M. Jain, A. 2002. *Unsupervised Learning on Finite Mixture Models.* IEEE transactions of pattern analysis and machine intelligence (Vol 24, no. 3).

[24] Haario, H., Saksman, E., Tamminen, J. 1999. *Adaptive proposal distribution for random walk Metropolis algorithm*. **Computational Statistics** (Vol 14, No 3). pp. 375-395. Available at: http://citeseer.ist.psu.edu/haario99adaptive.html.

[25] Verbeek, J., Nunnink, J. Vlassis N. 2006. *Accelerated EM-based clustering of large data sets*. **Data Mining and Knowledge Discovery** (Vol. 13/3). pp. 291-307. Available as PDF at: http://www.science.uva.nl/ṽlassis/pub/Verbeek04dmkd_rev.pdf.

[26] Gelman, A. 2004. *Prior distributions for variance parameters in hierarchical models*. **EconWPA, Econometrics** (Number 0404001). Available at: http://ideas.repec.org/p/wpa/wuwpem/0404001.html.

[27] Carroll, R., Ruppert, D., Stefanski, L. Crainiceanu C. 2006. *Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition*. London, Great Britain: Chapman Hall. 488 p. Chapter 9. Available as PDF at: http://www.stat.tamu.edu/čarroll/eiv.SecondEdition/bayesian.pdf. ISBN 1-584-88633-1

[28] Haario, H., Laine, M., Mira, A. Saksman E. 2003. *DRAM: Efficient adaptive MCMC*. **Reports of the Department of Mathematics, University of Helsinki** (preprint 374). Helsinki, Finland: Helsinki University. Available as PDF at: http://mathstat.helsinki.fi/reports/Preprint374.pdf.

[29] Haario, H., Saksman, E., Tamminen, J. 2001. *An adaptive Metropolis algorithm*. **Bernoulli** (Vol. 7(2)). pp. 223-242. Available at: http://citeseer.ist.psu.edu/haario98adaptive.html.

[30] Gelman, A., Roberts, G. Gilks W. 1996. *Efficient Metropolis jumping rules*. **Bayesian Statistics** (Vol. 5). pp. 599-608. Oxford University Press.

[31] Green, P. Mira A. 2001. *Delayed Rejection in reversible jump Metropolis-Hastings*. **Biometrika, 2001** (Vol. 88). pp. 1035-1053.

[32] Mira, A. 2001. *On Metropolis-Hastings algorithms with delayed rejection*. **Metron, 2001** (Vol. LIX, No. 3-4). pp. 231-241.

[33] Andrieu, C., Freitas, N., Doucet, A. Jordan, M. 2001. *An Introduction to MCMC for Machine Learning*. **Machine Learning** (Vol. 50). pp. 5-43. Available at: http://citeseer.ist.psu.edu/andrieu03introduction.html.

[34] Moral, D., Doucet, A. Jasra A. 2000. *Sequential Monte Carlo Samplers*. Technical Report, CUED/F-INFENG/TR 443. Cambridge University, Department of Engineering. Available at: http://citeseer.ist.psu.edu/moral02sequential.html.

[35] Celeux, G., Marin, J. Robert, C. *Iterated Importance Sampling in missing data problems*. Submitted to Computational Statistics and Data Analysis (to appear). Available as PDF at: http://www.ceremade.dauphine.fr/x̃ian/cmr03.pdf.

[36] Cappe, O., Guillin, A., Marin, J. Robert C. 2004. *Population Monte Carlo*. Available at: http://citeseer.ist.psu.edu/569964.html.

[37] Iba, Y. 2001. *Population Monte Carlo algorithms.* **Transactions of the Japanese Society for Artificial Intelligence** (Vol. 16, No. 2). Available as PDF at: http://www.ism.ac.jp/ĩba/journal.pdf.

[38] Douc, R., Guillin, A., Marin, J., Robert C. 2005. *Minimum variance importance sampling via Population Monte Carlo.* Available as PDF at: http://www.ceremade.dauphine.fr/x̃ian/dgmr05.pdf.

[39] Ridgeway, G. Madigan, D. 2002. *A Sequential Monte Carlo Method for Bayesian Analysis of Massive Datasets*. Netherlands: Kluwer Academic Publishers. Available at: http://citeseer.ist.psu.edu/541343.html.

[40] Vahteristo, K., Laari, A., Haario, H. Solonen, A. 2006. *Estimation of kinetic parameters in esterification of neopentyl glycol with propionic acid.* To be published. Lappeenranta. Finland: Lappeenranta University of Technology, Department of Chemical Engineering.

[41] Lilja, J. 2005. *A Fibrous Polymer-Supported Sulphonic Acid Catalyst in Esterification Processes*. PhD thesis. Turku, Finland: Laboratory of Industrial Chemistry, Process Chemistry Centre, Åbo Akademi. Article 3.

[42] Wärnå, J.m Geant, M., Salmi, T., Hamunen, A., Orte, J., Hartonen, R. Murzin, D. 2006. *Modeling and Scale-up of Sitosterol Hydrogenation Process: From Laboratory Slurry Reactor to Plant Scale*. **Industrial Engineering Chemistry Research** (Vol. 45, No. 21). pp. 7067-7076.

Appendix 1. Mathematical Definitions

Here some of the mathematical terms and notations are explained in more detail.

Convergence of Random Variables

In probability theory, several different types of convergence of random variables are defined. The weakest form of convergence is convergence $in\ distribution$. If we consider a sequence of random variables $(X_1, X_2, ...)$ and the corresponding sequence of CDF:s $(F_1, F_2, ...)$, we say that X_n converges in distribution to X (that has CDF F) if

$$\lim_{n\to\infty} F_n(a) = F(a) = P(X \leq a)$$

for every $a \in \mathbb{R}$ where F is defined and continuous. That is, the probability that X_n is in some given range gets arbitrary close to the probability that X is in the same range, when a very large n is chosen. Convergence in distribution is used in the Central Limit Theorem and in the Weak Law of Large Numbers. One result related to convergence in distribution is that if $X_n \to X$ in distribution, then $g(X_n) \to g(X)$ in distribution ($g : \mathbb{R} \to \mathbb{R}$).

Another term related to convergence in this work is convergence $almost\ surely$. The sequence X_n converges almost surely towards random variable X, if

$$P\left(\lim_{n\to\infty} X_n = X\right) = 1$$

which means that events that don not converge to X have probability 0. In general, a proposition is said to hold almost surely if $P(\text{"}proposition\ holds\text{"}) = 1$. Convergence almost surely implies also convergence in distribution. The notation is used in the Strong Law of Large Numbers.

Another type of convergence, namely convergence $in\ probability$ appears often in probability theory literature. The sequence X_n is said to converge to X in probability, if

$$\lim_{n\to\infty} P\left(|X_n - X| \geq \epsilon\right) = 0.$$

The definition is used in the weak law of large numbers in section 3.3.1.

CONTINUES

Mean, Variance, Covariance and Correlation

The expectation μ of a random variable X describes the most probable value of all possible values. The expectation is defined as (discrete and continuous case respectively)

$$\mu = E(X) = \sum_i x_i p(x_i),$$

$$\mu = E(X) = \int_{-\infty}^{\infty} x p(x) dx.$$

The variance of a random variable describes the variation and diffusion of the variable around its expectation. The variance for a discrete random variable with probability density function p is defined as

$$\sigma^2 = Var(X) = E(X - \mu)^2 = \sum_i (x_i - \mu)^2 p(x_i).$$

Similarly for continuous random variable with density function p:

$$\sigma^2 = Var(X) = E(X - \mu)^2 = \int_{-\infty}^{\infty} (x - \mu)^2 p(x) dx.$$

Using the Steiner's theorem we get a formula for variance that is easier to compute. Below it is given for both discrete and continuous case respectively.

$$\sigma^2 = \sum_i x_i^2 p(x_i) - \mu^2,$$

$$\sigma^2 = \int_{-\infty}^{\infty} x^2 p(x) dx - \mu^2.$$

The variance is seldom calculated analytically. Instead, a "sample variance" is often calculated from the measured data, using an estimate for the variance. If $x_k = (x_{1k}, ..., x_{nk})$ denotes the column vector in a design matrix (n observations for a variable), the sample variance can be calculated using

$$Var(x_k) = \sigma_{x_k}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ik} - \overline{x_k})^2.$$

The square root of the variance is the $standard\ deviation$ of a random variable:

$$Std(x_k) = \sigma_{x_k} = \sqrt{Var(x_k)}.$$

CONTINUES

The covariance is a measure of the relation (co-movement) between two random variables X and Y. The covariance is defined as

$$Cov(X,Y) = E((X - E(X))(Y - E(Y))).$$

If we look at the observation matrix, the corresponding covariance between two measured variables $x_l = (x_{1l}, ..., x_{nl})$ and $x_k = (x_{1k}, ..., x_{nk})$ can be calculated using

$$Cov(x_l, x_k) = \sigma_{x_l x_k} = \frac{1}{n-1} \sum_{i=1}^{n} (x_{il} - \overline{x_l})(x_{ik} - \overline{x_k}).$$

Note that $Cov(x,x) = Var(x)$. The covariance matrix $cov(X)$ where X is the observation matrix, is defined by $Cov(X) = [Cov(x_i, x_j)]$, where $i, j \in Z$ and $1 \leq i \leq p,\ 1 \leq j \leq n$. That is, the diagonal contains the sample variances and the other parts of the matrix consist of covariances between different measured variables.

The correlation coefficient between two random variables is defined as

$$Corr(x_l, x_k) = \rho_{x_l x_k} = \frac{\sigma_{x_l x_k}}{\sigma_{x_l} \sigma_{x_k}} = \frac{Cov(x_l, x_k)}{Std(x_l) Std(x_k)}.$$

That is, the non-diagonal elements of the covariance matrix can also be written as $\sigma_{x_i x_j} = \rho_{x_i x_j} \sigma_{x_i} \sigma_{x_j}$.

Confidence Intervals

If we are estimating some quantity θ, the $(1-\alpha) * 100\%$ confidence interval for it is the interval to which the true value of the parameter θ belongs with probability $1 - \alpha$. The principle of forming the confidence interval is simple: we assume that the estimated parameter θ follows some distribution D_θ. From D_θ we can assign the limits a adn b so that

$$P(a \leq D_\theta \leq b) = 1 - \alpha.$$

From the two inequalities $a \leq D_\theta \leq b$ it is possible to compute the limits for the parameter θ:

$$L \leq \theta \leq U.$$

For example if we assume that $X \sim N(\mu, \sigma^2)$, through CLT we know that $\overline{X} \sim N(\mu, \sigma^2/n)$ and thus

$$Z = \frac{\overline{X} - \mu}{\sigma^2/\sqrt{n}} \sim N(0,1).$$

CONTINUES

Now the $1-\alpha$ confidence interval for μ, for example, can be calculated from the equation

$$-z_{1-\alpha/2} \leq Z \leq z_{1-\alpha/2}$$

where z_x represents the point of the CDF of $N(0,1)$ at which $P(X \leq z_x) = x$.

Linear Regression Models

If the model is linear with respect to the unknown parameter θ, it can be written in the form $y = X\theta + \epsilon$. The estimate $\hat{\theta}$ that minimizes the LSQ function, is the solution of the normal equation $X^T X\hat{\theta} = X^T y$, which leads to

$$\hat{\theta} = (X^T X)^{-1} X^T y.$$

If the measurement error $\epsilon \sim N(0, \sigma^2 I)$ (i.i.d. components), we get $Cov(\hat{\theta}) = \sigma^2 (X^T X)^{-1}$.

Proof. Let $\hat{b}$ be the LSQ solution to the linear problem $y = Xb$. That is, $\hat{b} = (X^T X)^{-1} X^T y$ (normal equation). Now $Cov(\hat{b}) = Cov((X^T X)^{-1} X^T y)$. Using the assumption $Cov(\epsilon) = Cov(y) = \sigma^2 I$, the fact that $Cov(Ay) = ACov(y)A^T$ and $(AB)^T = B^T A^T$ and $(A^T)^{-1} = (A^{-1})^T$ ([8]) we get

$$\begin{aligned} Cov(\hat{b}) &= (X^T X)^{-}1 X^T \sigma^2 ((X^T X)^{-}1 X^T)^T \\ &= \sigma^2 (X^T X)^{-}1 (X^T X)(X^T X)^{-}1 \\ &= \sigma^2 (X^T X)^{-1}. \end{aligned} \tag{10.1}$$

In nonlinear regression, no exact distribution theory like the one presented above can be formed, and one has to rely on numerical methods.

Jacobian Matrix

The Jacobian matrix contains the partial derivatives of first order with respect to every function and every variable. If $y = y(x)$ is defined as a set of functions

$$\begin{aligned} y_1 &= y_1(x_1, ..., x_k) \quad ... \\ y_n &= y_n(x_1, ..., x_k) \end{aligned} \tag{10.2}$$

CONTINUES

then

$$J = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_k} \\ \cdots & & \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_k} \end{pmatrix}.$$

Hessian Matrix

The Hessian matrix contains the information about the second order (partial) derivatives of a certain function with respect to every variable pair. If $y = f(x_1, ..., x_n)$, the Hessian matrix is defined as

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}.$$

Quartiles and Interquartile Range

If we have ordered data with $2n$ observations, the first quartile q_1 is defined as the *median of the n first (smallest) observations*. Respectively, the third quartile q_3 is defined as the *median of the n largest observations*. The second quartile q_2 is the median of the whole data.

The interquartile range is used as a measure of spread in statistical samples. It is defined as $iq = q_3 - q_1$.

Principal Component Analysis

PCA (Principal Component Analysis) is meant for searching the directions, where the deviation of the samples is the largest. That is, we try to find the most important directions (directions that mostly describe the structure of the data) from a high-dimensional data set.

Numerically the directions of the greatest deviation can be obtained through the singular value decomposition (SVD) of the data matrix. The SVD for the observation matrix X (n x p) would be $X = U\Sigma V^T$ where U and V are orthogonal and $\Sigma = diag(\sigma_1, ..., \sigma_p)$. In this work the PCA techniques are considered as a way to rotate the proposal distribution in the single-component versions of the MCMC algorithms (SC and SCAM).