UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MODEL-DATA SYNTHESIS IN TERRESTRIAL ECOSYSTEM MODELING: INVERSE ANALYSIS AND INVERSE UNCERTAINTY ANALYSIS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

TAO XU

Norman, Oklahoma

2005

Downloaded from SHAREOK. Persistent Handle URL: https://hdl.handle.net/11244/862

UMI Number: 3163318



UMI Microform 3163318

Copyright 2005 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

MODEL-DATA SYNTHESIS IN TERRESTRIAL ECOSYSTEM MODELING:

INVERSE ANALYSIS AND INVERSE UNCERTAINTY ANALYSIS

A Dissertation APPROVED FOR THE
DEPARTMENT OF MATHEMATICS

BY

_____
Dr. Luther White

_____
DR. Yiqi Luo

_____
Dr. Kevin Grasse

_____
Dr. Semion Gutman

_____
Dr. S. Walter Wei

c Copyright by TAO XU 2005
All Rights Reserved

ACKNOWLEDGEMENTS

I am very grateful to my major advisors, Dr. Luther White and Dr. Yiqi Luo, for their invaluable advice and resourceful ideas that they gave to me during this work. I also thank them for their encouragement and support in guiding me into the field of ecosystem modeling.

I would like to express special thanks to Dr. Semion Gutman, Dr. Kevin Grasse and Dr. Walter Wei for their advice and help in various ways. In particular, I sincerely thank Dr. Semion Gutman for his supervision on my teaching when I served as a teaching assistant of his class, I thank Dr. Kevin Grasse for helping me understanding stochastic differential equation theory, and I thank Dr. Walter Wei for all inspiring talks and many helps in study and in life.

As always, my big family well deserves my love and thanks.

TABLE OF CONTENTS

CHAPTER 1 Introduction ……. 1

CHAPTER 2 Model description and inverse problem formulation …….. …6

2.1 Model description ……..9

2.1.1 State-space model ……9

2.1.2 Mapping from the state space to the data space …… ……. 11

2.1.3 Problem formulation …… ….. 11

2.2 Methods used in the study …… …….. 15

2.3 A practical carbon sequestration model …….. .. . 16

CHAPTER 3 Deterministic optimization ……. 21

3.1 Derivation of the gradient vector $\nabla J(c)$ ……. 22

3.1.1 Finite difference method for the model equation ……... …22

3.1.2 First derivative of the cost function $J(c)$ ……. . 24

3.2 Hessian matrix of the cost function $J(c)$ …… 29

3.3 Minimization algorithms …… . 40

3.3.1 Constrained steepest descent optimization method …… 40

3.3.2 Constrained conjugate gradient optimization method …… 42

3.4 Numerical results ……..... 42

CHAPTER 4 Stochastic optimization ……..... 46

4.1 GA Description …… 47

4.1.1 General description of genetic algorithm …….. 47

4.1.2 Specifications of GA with respect to Model (2.1) ~ (2.6) …….. 50

4.1.3 Numerical result using GA …… 53

4.1.4 Discussion of the numerical result …… 55

4.2 Simulated annealing (SA) …….. 56

4.2.1 General description of SA ……. 56

4.2.2 Numerical result …… 60

CHAPTER 5 Bayesian statistical inference and parameter estimation …….. 64

5.1 Forward/inverse probability and Bayes' Theorem ……. 66

5.2 Proposing the inverse probability density function of Model (2.1) …… 67

5.2.1 The prior information ……68

5.2.2 The posterior probability density function (PPDF) $p(c)$ …… ...69

5.3. An approximation approach for inverse uncertainty analysis …….. 70

5.4 A stochastic simulation approach for inverse uncertainty analysis

– Markov chain Monte Carlo (MCMC)……... 72

5.4.1 Markov chain …… 73

5.4.2 Stationary distribution …….. 74

5.4.3 Metropolis-Hastings (M-H) MCMC …….. . 75

5.4.4 MCMC estimator and output analysis …….. 77

5.5 Toward a more efficient sampling method …….. . 79

5.5.1 Symmetric proposal probability …… ……..... 79

5.5.2 Stationary proposal probability …… …….. 80

5.5.3 The proposal probability that increases the sampling efficiency …… 81

5.6 Application to the seven-pool model and numerical results …… 83

5.6.1 Numerical result of the analytical approach …… 83

5.6.2 Numerical result of the MCMC approach with the standard M-H algorithm …… 92
5.6.2.1 Specifications of the MCMC sampling using the standard M-H algorithm …… 92
5.6.2.2 Output analysis …….. 96
5.6.3 MCMC with fast sampling method …….. 96
5.6.3.1 Specifications of the MCMC sampling ……. 97
5.6.3.2 Output analysis …….. 100
5.7. Result comparison …… 100
5.8 Data comparison …… 102
5.9 Model prediction …… 105
CHAPTER 6 CONCLUSIONS AND FURTHER WORK …… 108
REFERENCES …… 112

ABSTRACT

In this thesis we investigate various inversion approaches for a general type of biogeochemical cycle model that describes the carbon sequestration mechanism of terrestrial ecosystem. We formulate the inverse problems in two approaches – a deterministic inverse approach and a probabilistic inverse approach. We first develop the deterministic inverse techniques by calculating the first and second derivatives of the cost functional. Algorithms that depend on the gradient information are proposed. Then, considering the stochasticity in the model, we introduce two stochastic optimization methods – genetic algorithm and simulated annealing, to estimate the model parameters. We further consider the inverse uncertainty of the problem and introduce the Bayesian paradigm to formulate a posterior probability density function that describes the inverse uncertainty. Function approximation approach and Markov Chain Monte Carlo technique are then used to study the probability density function to reveal the inversion result. To increase the simulation efficiency, we combine Hessian matrix information with the proposal probability density function in the Metropolis-Hastings algorithm for fast sampling. All the approaches are tested against a practical numerical model that describes forest ecosystem carbon sequestration. The thesis concludes by comparing the various approaches and by discussing further issues that needs to be studied.

CHAPTER 1

INTRODUCTION

The rapid increase in atmospheric carbon dioxide (CO_2) concentration provides an urgent need to quantify potential carbon sinks in terrestrial ecosystems. In the past two decades, more than a dozen biogeochemical models have been developed to predict terrestrial carbon sequestration in response to increasing atmospheric CO_2 (e.g., Parton et al. 1987; Rastetter et al. 1997; Comins and McMurtrie 1993; Mellillo et al. 1993; Luo and Reynolds 1999; Thompson and Randerson 1999). Most of those models share a common compartmental structure that partitions photosynthetically fixed carbon into several pools. The number of carbon pools in each model may vary according to different ecosystem types, for example, two-pool model (Rastetter et al. 1997) verses twelve-pool model (Thompson and Randerson 1999), seven-pool model (Luo et al. 2003) verses twelve-pool model (Luo and Reynolds 1999), but the general compartmental model structure is the same. The capacity of ecosystem carbon sequestration is strongly related to the residence times of carbon in these pools. Inverses of the residence times are parameters called the

carbon transfer rates that indicate carbon decomposition speeds of various carbon pools. It is necessary to know these carbon transfer rates from observational data sets to quantify the carbon sequestration potential of a terrestrial ecosystem under the constantly changing atmospheric CO_2. This research is focused on applying and developing both deterministic and stochastic inversion techniques to solve the inverse problem of the general type of biogeochemical cycle models that describe terrestrial ecosystem carbon cycles. The motivation of the study is mainly based on the following:

1. Global warming and carbon cycle study is becoming increasingly important. With the accumulation of observational data and the development of various model types, there is an increasing recognition that one should combine terrestrial carbon observational data and process models in systematic ways.
2. Parameter estimation can be a formidable task given the complexity of the ecosystem models and the often limited amount of data, though ecologists' prior knowledge can be utilized to constrain model parameters, the prior information is usually not sufficient to accommodate multiple observational data sets.
3. The randomness in the model functions and the non-linearity of objective functions with respect to parameters often mean multiple optimal solutions. In such a case, it is necessary to introduce both local and global optimization techniques.
4. Ecosystem measurement is subject to uncertainty that affects parameter estimation and correspondingly the model projection. It is being realized by the global change research community that data uncertainty is an integral part of

observation and is as important as data values themselves from the model-data synthesis standpoint and the prediction standpoint. Thus uncertainty study poses an indispensable component in this thesis. It is necessary to introduce probabilistic approaches that not only give parameter estimations but also estimation uncertainties.

The goal and also the contribution of the study are mainly fivefold:

1. To formulate a general type of terrestrial ecosystem inverse problems and develop inverse solutions from both a deterministic approach and a stochastic approach, and test the techniques with a practical carbon model.
2. To apply probabilistic inversion technique and study the information content of the observational data sets on model parameters. The study covers parameter estimation, uncertainty estimation, parameter identifiability and prediction uncertainty simulation. Two approaches for studying the inverse uncertainty, an analytical approach using function approximation and a simulation approach using Markov chain Monte Carlo (MCMC), are developed. Both approaches reveal the information content of the posterior probability density function (PPDF) of the model parameters.
3. To develop an efficient algorithm that reduces computational cost. MCMC simulation of ecosystem model over a long time span can be very time-consuming. It is always a good practice to introduce fast algorithms to improve the MCMC sampling efficiency whenever possible. In the study, a Metropolis-

Hastings (M-H) type of algorithm is coupled with the Hessian matrix information of the cost function to greatly increase the sampling efficiency.

4. To compare what these various approaches reveal by solving a state-of-art biogeochemical model describing the carbon cycle of forest ecosystem. The interesting aspect of the model is that the data sets cannot completely determine the model parameters and is thus ill-conditioned. The comparison will make clear how the different approaches reveal the same fact, but from different perspectives.
5. Inverse problems are intensively studied and applied in various mathematical and engineering fields, but their application to the ecological study is still in an early stage. By synthesizing various approaches in the context of an ecosystem model inverse problem, we hope to build a platform for data-model synthesis of general types of biogeochemical models to benefit future research of this area.

The organization of the paper is the following. In Chapter 2, we give an introduction of the general inverse problem, as well as the ecosystem carbon model description. We then formulate the inverse problem in both a deterministic and a stochastic approach. We also introduce a practical model describing forest ecosystem carbon sequestration. This model will be used for numerical testing throughout the thesis. In Chapter 3, we explore the deterministic optimization approach by developing the Jacobian and the Hessian matrices for the cost function of the general model type. A steepest-descent-type of algorithm and a conjugate-gradient-type of algorithm are introduced. One algorithm is then applied to the practical numerical model to search for an inverse solution. The implication of the inversion result is discussed. Considering the possible multi-optima caused by

stochasticity, in Chapter 4 we introduce two global optimization techniques, genetic algorithm (GA) and simulated annealing (SA), and apply both of them to the model with numerical testing. In Chapter 5, considering the uncertainty in the observational data sets, we introduce the Bayesian framework and formulate the inverse problem as a problem of determining the joint probability of the model parameters. We propose a posterior probability density function (PPDF) of the model parameters under the assumption that the observational data has Gaussian type of random errors. Two methods to analyze the information in the PPDF are then introduced: one is the Laplace approximation approach, which is to approximate the PPDF by a Gaussian function; the other approach is the Markov chain Monte Carlo (MCMC) simulation, which is to sample the PPDF directly using the Metropolis-Hastings (M-H) type of algorithm. Both approaches reveal the parameter value information, the parameter correlation information and the estimation uncertainty. Applications to the numerical model are made and the results are compared. Along the discussion in Chapter 5, we also propose an efficient sampling algorithm that utilizes the Hessian matrix information to tremendously increase the sampling efficiency and thus reduce the computational cost of forward simulation. In Chapter 6, summarization and conclusions about the various inverse approaches are made, and issues for further study are listed.

CHAPTER 2

MODEL DESCRIPTION AND INVERSE PROBLEM FORMULATION

Inverse problems are widely studied in applied mathematics, seismology, oceanography, atmosphere science and other engineering and science fields (e.g., Tarantola, 1987; Hensel, 1991; Enting, 2002). They are defined as problems of finding the cause of an observed effect, in that "the chain of calculation or inference was in the opposite direction to real-world causality" (Enting, 2002). An inverse problem is always coupled with a forward problem that provides the effect of a given cause. Generally, an inverse problem is related to an operator equation: $z = F(c)$ with $F : \Omega \to Z$ being a possibly nonlinear operator between two metric spaces Ω and Z. The forward problem is to determine the effect of a given cause, whereas the inverse problem is to determine the cause given the effect. The space Ω is commonly denoted the model space or the

parameter space, which is assumed as a finite dimensional space in this thesis. The space Z is denoted as the data space. If F is a perfect model of a physical process and z is some data quantity without any error, then the system is perfectly deterministic. However, in reality $z = F(c)$ is unlikely to hold for a practically measured quantity z, since measurements may have finite precision or errors. In addition, the model may be inaccurate in the sense that the operator F may not model all aspects of the physical processes that produce the observations. With these considerations in mind, the model should be more realistically written as $z = F(c) + e$ with e being an error term describing observation error and possibly model error as well.

The problem of finding c given z or rather given the pair $\{z, e\}$ is called an inverse problem. Inverse problem can be classified as well-posed or ill-posed. A problem is well-posed if there exists a unique, stable solution. In practice, it is commonly true that the forward problems are well-posed while the corresponding inverse problems are ill-posed.

The following definition of a well-posed inverse problem is attributed to Hadamard (1923):

Definition. (Well-posed inverse problem). Let Ω and Z be two normed spaces and let $F : \Omega \to Z$ be a continuous operator. The inverse problem of $Z = F(c)$ is well-posed in the sense of Hadamard if the following three conditions hold:

1. Existence: there is a solution $c \in \Omega, \forall z \in Z$ with $F(c) = z$.
2. Uniqueness: There exists at most one solution $c \in \Omega$ for any $z \in Z$.
3. Stability: $\forall \varepsilon > 0, \exists \delta(\varepsilon)$ so that $\| F(c_1) - F(c_2) \| < \delta(\varepsilon)$ whenever $\| c_1 - c_2 \| < \varepsilon$.

Problems for which at least one of the three conditions above fails to hold are called ill-posed. Whether a problem is well-posed or ill-posed depends on the operator $F : \Omega \to Z$ as well as the spaces Ω and Z. In practice, it is rare that an inverse problem is well-posed. Ill-posed inverse problems are more common due to the incomplete data information, the choice of admissible set and model structures.

In particular, for ecosystem inverse problem, the description for parameter-estimating procedures can be phrased in the following way: given a relation $F(c) = z$, where F is an general operator representing an ecosystem model in the form of a set of equations (linear equations, ordinary differential equations or partial differential equations) or simply a function, c represents either an unknown function or a set of parameters, and z is some observation (if c is an unknown function of a finite-parametric family of functions, then $c = c(t) = c(c_1, c_2, ..., c_m, t)$ where $(c_1 \quad . \quad . \quad c_m)$ is a vector of parameters and t is an independent variable), then the question is to find parameters $c = (c_1 \quad . \quad . \quad c_m)$ by solving the minimization problem:

$$c \in \arg\{\min[J(F(c) - z)]\}$$

where J is a positive functional, $c \in \Omega$ where Ω is an admissible set of c.

Depending on the mapping F, the parameter space and the data space, the above minimization problem may have a unique solution (for example certain continuity and convexity conditions of J and the compactness condition of the admissible set Ω are met).

In many situations J may have several local solutions and global ones over Ω (for example if the convexity of J over Ω is violated). Various local and global techniques such as the descent-type-of algorithms, the global optimization techniques (simulated annealing, genetic algorithm, tabu search, downhill-simplex method, etc.) are invented to handle different situations. Moreover, in ecosystem modeling and observation, due to the complexity of the actual ecosystem and the imperfection of the mathematical models and the measurement methods, modeling uncertainty and observational uncertainty always exist. It is common that observation z contains relatively large errors so that instead of some "true" observation z, one has observation $z_e = z + e$. How is the probabilistic perturbation of data reflected in the parameter estimation? To address such a question, one needs to follow a probabilistic approach.

Based on the above scenarios we will discuss different effective approaches respectively in this study. But first we give the description of the general terrestrial carbon cycling model, the inverse problem formulation and a practical seven-pool carbon model parameterized with actual observational data sets.

2.1 Model description

2.1.1. State-space model

The general mathematical model describing carbon cycle of a terrestrial ecosystem is given by the following system of differential equations (Luo et al. 2003; White et al. 2002):

$$\begin{aligned} &\frac{dX(t)}{dt} = \xi(t)ACX(t) + BU(t) \\ &X(0) = X_0 \end{aligned} \tag{2.1}$$

where,

1. $X = (x_1 \quad x_2 \quad ... \quad x_m)^T$ is a $m \times 1$ vector describing carbon pool sizes.
2. A is an m×m matrix describing the proportional carbon distribution among various carbon pools. It is taken as a constant matrix in the study.
3. C = diag (c) is an $m \times m$ diagonal matrix with the diagonal elements given by the carbon transfer coefficients: $c = (c_1, c_2, ..., c_m)^T$.
4. B is an $m \times 1$ vector that partitions the system input.
5. $\xi(.)$ is a real valued function given as a time series with prominent seasonal periodicity to simulate the seasonal modulation effect on the carbon decomposition mechanism. The function is mainly determined by seasonal moisture and temperature variation. Usually $\xi(.)$ is contaminated with disturbances.
6. $U(.)$ is a system input function given as a time series modeling carbon input into the system. Usually $U(.)$ is contaminated with disturbances as well.
7. X_0 is an initial condition vector.

2.1.2 Mapping from the state space to the data space.

Assume there are *s* data sets $\breve{Z}^k, k = 1,2...,s$ in the observation. Corresponding to the *s* data sets, we have the following observation operators: $\phi^k, k = 1,2,...s$ with each of the ϕ^k's taking the general form of

$$\phi^{k^T} = c^T H^k + \varphi^{k^T}, k = 1,2,...s \tag{2.2}$$

where H is an $m \times m$ constant diagonal matrix and φ^k is an $m \times 1$ constant vector. T represents transpose of vector.

2.1.3 Problem formulation.

We write the objective function in a discrete form for the convenience of numerical analysis. Approximation errors between the continuous model and its discrete version are neglected in the study, although the errors do propagate and influence the assumed observation errors and is an interesting question in its own right. Let $\widetilde{X}(c)$ denote the vector of state variables at all time steps from a finite difference method scheme that solves the equation (2.1) numerically. Let $\Phi^k(c)$ denote the mapping matrix from $\widetilde{X}(c)$ to data set $\breve{Z}^k, k = 1,2,...,s$ (with a slight abuse of notation, $\breve{Z}^k, k = 1,2,...,s$ also denotes the vector consisting of the observation time series of the *k*-th data set). Let the error covariance matrix for the *k*-th data set be $\mathrm{cov}(E^k)$. Assume the errors of different data sets are independent. Then the cost function is formulated as:

$$J(c)=\sum_{k=1}^{s}\frac{1}{2}\left(\breve{Z}^{k}-\Phi^{k}(c)\widetilde{X}(c)\right)^{T}\left(\mathrm{cov}(E^{k})\right)^{-1}\left(\breve{Z}^{k}-\Phi^{k}(c)\widetilde{X}(c)\right) \tag{2.3}$$

However, there are situations where the data sets used may not give us complete information about the parameter *c*, as in the case of an ill-conditioned inverse problem. It is necessary to include regularization scheme to allow the prior selection of some parameter from a feasible set of parameters. The most common and well-known regularization scheme is the Tikhonov (Tikhonov 1963) regularization. By introducing the Tikhonov regularization, we form the following weighted sum of data information and prior information about the parameter:

$$J(c)=\sum_{k=1}^{s}\frac{1}{2}\left(\breve{Z}^{k}-\Phi^{k}(c)\widetilde{X}(c)\right)^{T}\left(\mathrm{cov}(E^{k})\right)^{-1}\left(\breve{Z}^{k}-\Phi^{k}(c)\widetilde{X}(c)\right)+\frac{1}{2}\lambda\left\langle G(c-c^{prior}),(c-c^{prior})\right\rangle \tag{2.4}$$

In the above, c^{prior} is some prior information about the parameter *c*, λ specifies the weight of the prior information, G is a linear operator in the form of a matrix. In applications, G is usually either the identity matrix or some approximation to the prior covariance matrix of the parameter *c*. Depending on specific problems, λ can be chosen as a sequence that tends to zero in iterative optimization techniques, or it can be chosen based on the "L-curve" (see [Hansen] for a detailed discussion about "L-curves" for regularization). But we will see later that there is a natural correspondence between the choice of the regularization parameter λ and the setting-up of the Bayesian prior information.

If, in particular, we assume the observation errors within each data set $\breve{Z}^k, k = 1,2,...,s$ are also independent with variances $\sigma_k, k = 1,2,...,s$, then the off-diagonal elements in the covariance matrix $\mathrm{cov}(E^k)$ are zeros and the cost functional (2.4) can be written in the following simple quadratic form:

$$J(c) = \frac{1}{2}\sum_{k=1}^{s}\frac{1}{\sigma_k^2}\sum_{j=1}^{N_k}[\phi^k(c)^T X(c)(t_j^k) - Z_j^k][\phi^k(c)^T X(c)(t_j^k) - Z_j^k] + \frac{1}{2}\lambda < G(c - c^{prior}), c - c^{prior} > \tag{2.5}$$

where Z_j^k is the j-th observation of the *k*-th data set $\breve{Z}^k$ $(k = 1,...,s; j = 1,...,N_k), \phi^k(c)$ is the mapping vector of the *k*-th data set.

To formulate the optimization problem, let Ω denote be *the admissible set* for parameter *c* in R^m. The problem is to find $c^{op} \in \Omega$ such that

$$J(c^{op}) = \min\{J(c) \mid c \in \Omega\} \tag{2.6}$$

Solving problem (2.6) gives a set of optimal point values for parameter *c*. In ecosystem parameter estimation problems, it is reasonable to require the set Ω being compact to allow the existence of the optimal solution. In fact, one often has the case that $\Omega = \prod_{i=1}^{n} I_i$, where I_i are closed intervals defining the physically feasible ranges of the

parameters (e.g., White et al. 2002; Luo et al. 2003). In the case that $\xi(.)$ and $U(.)$ are modeled by Lipschitz continuous functions, it is easily seen the $J(c)$ is continuously differentiable and thus the existence of the optimal solutions (Rabenstein, 1972).

The above discussion formulated the inverse problem as *a least-squares optimization* problem. As was mentioned earlier, parameter estimation uncertainty always exists due to the uncertainty in the observational data. To study the estimation uncertainty of the parameters caused by the uncertainty in the data source, one needs to formulate the PPDF for the parameters. A systematic treatment of inverse problems from a probabilistic point of view can be found in Tarantola (1987). We will defer the precise formulation of the probabilistic inverse problem of Model (2.1) ~ (2.3) till Chapter 5 after we introduce the Bayesian inverse probability. But roughly, the probabilistic formulation can be phrased in the following manner: Let the model equation (2.1) and the observation operators (2.2) be combined such that the forward mapping is denoted by $F : c \in \Omega \to Z$, and let the observation be given by Z_e with $Z_e = F(c^{true}) + e$, where e represents the observational error information. Given the assumption that the error distribution is known, how to make statistical inference on the model parameter c?. Baye's theorem provides a theoretical answer. In Bayes' theorem, the posterior knowledge is combined with the prior knowledge, and the corresponding conditional probability then gives the inverse solution in a probabilistic manner.

2.2 Methods used in the study

Corresponding to the above problem formulations, we develop both deterministic and stochastic optimization techniques. The deterministic techniques are the gradient descent type of algorithms that are mostly used in nonlinear optimization problems. They include, for example, the steepest-descent method, the conjugate-gradient method, the quasi-Newton method, and the Levenberg-Marquardt algorithm (see Press (1992) for detailed discussions of the methods). Gradient-descent type of methods are mainly based on the vector $\nabla J(c) = \left(\partial J(c)/\partial c_i\right)_{i=1,\ldots m}$. Their advantages are the relative simplicity and low computational cost; the main disadvantage is that if the surface $J(c)$ has multiple minima over the admissible set Ω, the algorithms then tend to find a local minimum near the initial value of c rather than locate the global minimum. To find the desired minimum point, some prior knowledge about the global optimal c^{o} should be utilized to start searching: one either uses the Tikhonov regularization scheme or one simply starts from c^{prior} near to the global optimal and finds the first optimal point. The stochastic approaches such as the genetic algorithm, simulated annealing, downhill simplex method, tabu search, etc. are global search methods that tend to find the global optimal by searching the whole space Ω. They overcome the local-minimum pitfalls of gradient-descent methods, but have the disadvantage of higher computational costs. The stochastic approaches may be combined with gradient-descent methods for finding the global minimum with increased efficiency. For example, a good discussion of global-local hybrid optimization can be found in Goldberg (1999).

More generally, as was mentioned above, since inverse problems are probabilistic in nature, one should rather treat the parameters as random variables rather than deterministic values. Bayesian inversion gives a probabilistic description of the model parameters by packaging data information and prior knowledge in a posterior probability distribution function (PPDF). Once this distribution function is known, the only thing left is to study this function. One can then use different approaches such as Monte Carlo integration, function approximation and MCMC simulations to reveal the information content in the PPDF. We refer to Tarantola (1987) and Box & Tiao (1973) for a complete discussion of inverse problems from a probabilistic point of view and discussions of Bayesian analysis.

We will study the inverse problem formulated in (2.1) ~ (2.6) using the various approaches mentioned above. For numerical testing purpose, we introduce the following numerical model.

2.3 A practical carbon sequestration model

The following model describes the carbon sequestration mechanism of the Duke forest ecosystem with a structure shown in Figure 2.1, and is parameterized with data sets collected from the Duke forest Free Air CO_2 Enrichment (FACE) from the year of 1996 to the year of 2000 (Luo et al. 2003). The model structure is the same as (2.1) with $m = 7$ and the following corresponding matrices and initial conditions:

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0.712 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0.288 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0.45 & 0.275 & -1 & 0.42 & 0.45 \\ 0 & 0 & 0 & 0.275 & 0.296 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0.004 & 0.03 & -1 \end{pmatrix}.$$

$$B = (0.25 \quad 0.30 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)^T$$

$$X_0 = (469 \quad 4100 \quad 64 \quad 694 \quad 123 \quad 1385 \quad 923)^T$$

$$C = diag(c) \text{ with } c = (c_1, c_2, ..., c_7)^T$$

The two functions $\xi(.)$ and $U(.)$ are given by two time series shown in Figure 2.2. There are six data sets: woody biomass, foliage biomass, litter-fall, carbon in forest floor, carbon in forest mineral soil and soil respiration that can be used for the inverse problem study. Corresponding to the six data sets, there are six mapping operators:

$$\phi_1 = (0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$
$$\phi_2 = (0.75 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$
$$\phi_3 = (0.75c_1 \quad 0.75c_2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$$
$$\phi_4 = (0 \quad 0 \quad 0.75 \quad 0.75 \quad 0 \quad 0 \quad 0)$$
$$\phi_5 = (0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1)$$
$$\phi_6 = (0.25c_1 \quad 0.25c_2 \quad 0.55c_3 \quad 0.45c_4 \quad 0.7c_5 \quad 0.55c_6 \quad 0.55c_7)$$

The admissible set Ω is a closed set formed by the Cartesian product of the following intervals for each of the components of vector $c = (c_1, c_2, ..., c_7)^T$:

$$
\begin{aligned}
1.700\times10^{-4} &\le I_1 \le 2.950\times10^{-3} \\
5.480\times10^{-5} &\le I_2 \le 2.740\times10^{-4} \\
5.479\times10^{-3} &\le I_3 \le 2.734\times10^{-2} \\
5.480\times10^{-4} &\le I_4 \le 2.740\times10^{-3} \\
2.740\times10^{-3} &\le I_5 \le 6.850\times10^{-3} \\
5.480\times10^{-5} &\le I_6 \le 2.740\times10^{-4} \\
1.370\times10^{-6} &\le I_7 \le 9.130\times10^{-6}
\end{aligned}
\qquad (2.7)
$$

Throughout the numerical analysis in this study, the following standard deviations are assumed to normalize the six given data sets:

$$\sigma_1 = 1.18, \sigma_2 = 530, \sigma_3 = 50, \sigma_4 = 70, \sigma_5 = 35, \sigma_6 = 45 .$$

These standard deviations are mostly assumed to be proportional to the ranges of the given data sets. However, relative small standard deviations are assumed for slow SOM and passive SOM due to the limited amount of observations in these two data sets.

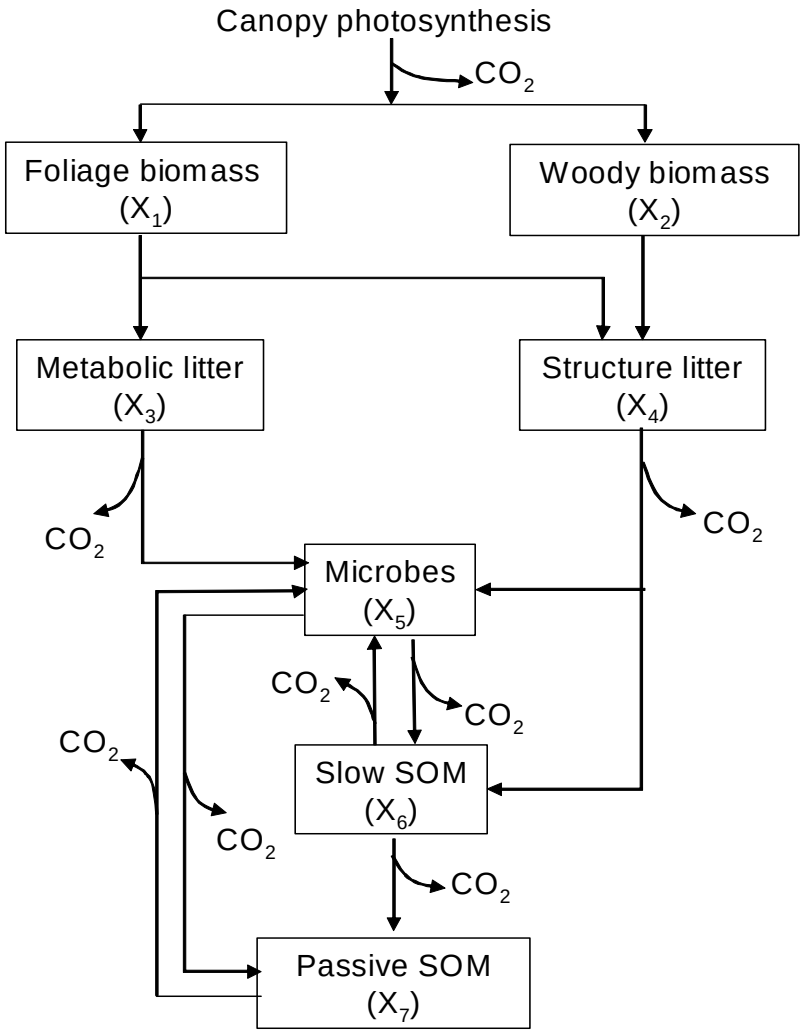

Figure 2.1. Structure of the seven-compartment model. There are seven carbon pools in the model, connected by carbon transfer coefficients that partition carbon distribution among various pools. (The figure was adopted from Luo et al. 2003 under author's permission.)

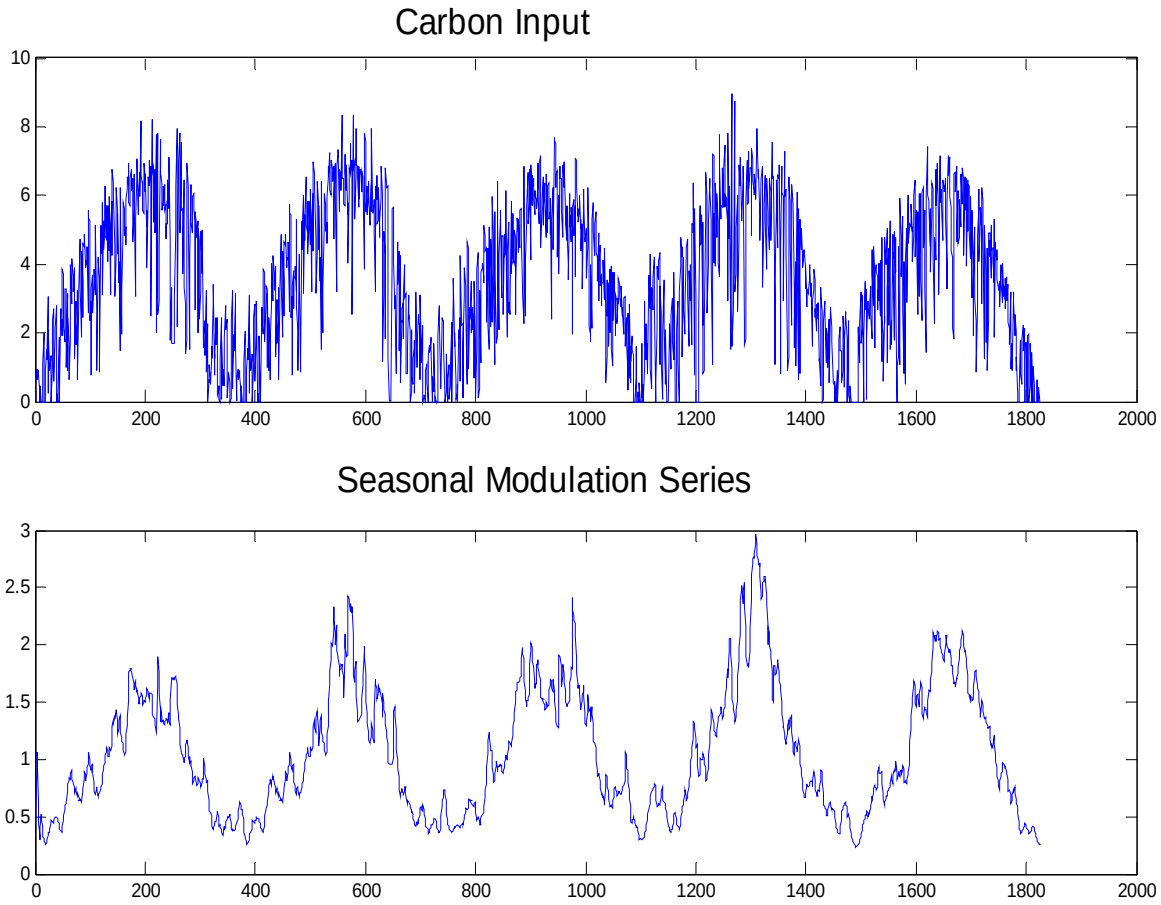

Figure 2.2. Time series in the numerical model (2.1). This figures shows the modeled carbon input $U(.)$ (above) by canopy photosynthesis, as well as the seasonal modulation function $\xi(.)$ (below) that models the combined effects of moisture and temperature on the carbon transfer rates. It can be see that both of them have jumps and random noise components, especially in $U(.)$. Such time series are very common in ecosystem observation.

CHAPTER 3

DETERMINISTIC OPTIMIZATION

In this Chapter, we develop a framework for deterministic optimization algorithms of model (2.1) ~ (2.6) by giving the gradient vector $\nabla J(c)$ and the Hessian matrix $\nabla^2 J(c)$. The gradient vector $\nabla J(c)$ is mainly used for giving descent direction. However, the Hessian matrix is derived for three purposes throughout the study:

1. To be used in the function approximation approach to study the posterior probability density function (PPDF) of parameters by approximating the PPDF using Gaussian function.
2. To be used for the proposal distribution for fast sampling of the PPDF in MCMC simulation.
3. To be used in the steepest-descent type of algorithms such as the Levenburg-Marquart algorithm, Newton's method etc. where the second derivative information is needed.

3.1 Derivation of the gradient vector $\nabla J(c)$

We first compute $\nabla J(c) = \left(\partial J(c)/\partial c_i\right)_{i=1,\ldots m}$. Similar derivation can be found in White et. al. (2002) for single observation data set (soil respiration). Here we derive the method under a slightly more general setting where the observation consists of multiple data sets and the prior information is included in the cost function. Throughout the derivation, $\xi_j = \xi(t_j)$, $X_j = X(t_j) = \left(x_1(t_j), x_2(t_j), \ldots, x_m(t_j)\right)^T$, $U_j = U(t_j)$, and $t_j^{\,k}$ is the *j*-th observation time of the *k*-th data set. To facilitate the computation, we introduce the adjoint equation similarly as in White et al. (2002).

3.1.1 Finite difference method for the model equation

Let $\left\{ t_j \right\}_{j=0}^{N}$ be an equally spaced partition of the time interval [0, T]. With finite-difference method, Equation (1) is solved numerically by the following steps:

$$X_{j+1} - X_j = [\theta\xi_{j+1} ACX_{j+1} dt + (1-\theta)\xi_j ACX_j dt] + B[\theta U_{j+1} dt + (1-\theta)U_j dt]$$
$$j = 0,1,2,\ldots N-1$$

When $j = 0$, we have

$$[I - \theta\xi_1 ACdt]X_1 = [I + (1-\theta)\xi_0 ACdt]X_0 + B[\theta U_1 dt + (1-\theta)U_0 dt]$$

When $j = 1, \ldots$ N-1, we have

$$-[I+(1-\theta)\xi_1 ACdt]X_1+[I-\theta\xi_2 ACdt]X_2 = B[\theta U_2 dt+(1-\theta)U_1 dt]$$
. . . . . .
$$-[I+(1-\theta)\xi_{N-1} ACdt]X_{N-1}+[I-\theta\xi_N ACdt]X_N = B[\theta U_N dt+(1-\theta)U_{N-1} dt]$$

The above can be written as an algebraic system:

$$\begin{pmatrix} B_1(c) & & & \\ -\hat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & -\hat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN} \breve{X}(c)_{mN\times 1} = \begin{pmatrix} \hat{B}_0(c)X_0(c) \\ 0 \\ \cdots \\ 0 \end{pmatrix}_{mN\times 1} + \breve{f}_{mN\times 1} \qquad (3.1)$$

where $B_j(c)=[I-\theta\xi_j ACdt], \hat{B}_j(c)=[I+(1-\theta)\xi_j ACdt], f_j = B[\theta U_j dt+(1-\theta)U_{j-1}dt].$

and

$$\breve{X}(c)_{mN\times 1} = \begin{pmatrix} X_1(c) \\ X_2(c) \\ \cdots \\ X_N(c) \end{pmatrix}_{mN\times 1}, \qquad \breve{f}_{mN\times 1} = \begin{pmatrix} f_1 \\ f_2 \\ \cdots \\ f_N \end{pmatrix}_{mN\times 1}$$

Set $\tilde{B}(c)_{mN\times mN} = \begin{pmatrix} B_1(c) & & & \\ -\hat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & -\hat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN}$, Equation (4) is then written as:

$$\widetilde{B}(c)\breve{X}(c) = \begin{pmatrix} \widehat{B}_0(c)X_0(c) \\ 0 \\ \cdots \\ 0 \end{pmatrix} + \breve{f} \tag{3.2}$$

Now we have derived the algebraic equation that numerically approximates the solution of its continuous version Equation (2.1).

3.1.2 First derivative of the cost function $J(c)$

We use cost function (2.5) to derive the derivative of $J(c)$**.** The cost function (2.5) can be written in a matrix form:

$$J(c) = \frac{1}{2}\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c) - \breve{Z}^k]^T[\Phi^k(c)P^k\breve{X}(c) - \breve{Z}^k] + \frac{1}{2}\lambda < G(c - c^{prior}), c - c^{prior} > \tag{3.3}$$

where

$$\Phi^k(c) = \begin{pmatrix} \phi^k(c)^T & & & \\ & \phi^k(c)^T & & \\ & & \cdots & \\ & & & \phi^k(c)^T \end{pmatrix}_{(N_k \times (mN_k))} \quad \text{and} \quad \breve{Z}^k = \begin{pmatrix} Z_1^k \\ Z_2^k \\ \cdots \\ Z_{N_k}^k \end{pmatrix}_{((mN_k)\times 1)}$$

are the matrices consisting the mapping operator vectors of the *k*-th data set and the observed data points of the *k*-th data set, respectively. ${P^k}_{mN_k \times mN}$ is a projection matrix that projects the space of the state variable at all time steps $\{t_j\}_{j=0}^{N}$ to the space of state variables associated with observation time steps $\{{t_j}^k\}_{j=1}^{N_k}$ of the *k*-th data set.

Take the derivative of the cost function with respect to c, we have:

$$
\begin{aligned}
DJ(c)c' &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T [(D\Phi^k(c)c')_{N_k \times mN_k} P^k \breve{X}(c)] \\
&+ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T [\Phi^k(c) P^k (D\breve{X}(c)c')_{mN \times 1}] + \lambda G(c - c^o)c' \\
&= (I) + (II) + (III)
\end{aligned}
\tag{3.4}
$$

with

$$
\begin{aligned}
(I) &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T (D\Phi^k(c)c')_{N_k \times mN_k} P^k \breve{X}(c) \\
(II) &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T \Phi^k(c) P^k (D\breve{X}(c)c')_{N \times 1} \\
(III) &= \lambda G(c - c^o)c'
\end{aligned}
$$

To compute (I), we need to compute $(D\Phi^k(c)c')$. Since the mapping operators are of the form $\phi^k(c)^T = c^T H^k + \varphi^{kT}$ in general, we have

$$
D[\phi^k(c)(t)^T]c' = c'^T H^k = \sum_{i=1}^{m} c_i' e_i^T H^k
$$

i.e.,

$$
(D\Phi_k(c)c') = c_1' \begin{pmatrix} e_1^T H^k & & \\ & \ddots & \\ & & e_1^T H^k \end{pmatrix} + \ldots + c_m' \begin{pmatrix} e_m^T H^k & & \\ & \ddots & \\ & & e_m^T H^k \end{pmatrix}_{(N_k \times mN_k)}
\tag{3.5}
$$

Now

$$
\begin{aligned}
(I) &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T (D\Phi^k(c)c') P^k \breve{X}(c) \\
&= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T \\
&\times \left(c_1' \begin{pmatrix} e_1^T H^k & & \\ & \cdot & \\ & & e_1^T H^k \end{pmatrix} + \ldots + c_m' \begin{pmatrix} e_m^T H^k & & \\ & \cdot & \\ & & e_m^T H^k \end{pmatrix} \right)_{(N_k \times mN_k)} \begin{pmatrix} X(t_1^k) \\ X(t_2^k) \\ \cdot \\ X(t_{N_k}^k) \end{pmatrix} \\
&= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T \left(c_1' \begin{pmatrix} e_1^T H^k X(t_1^k) \\ \cdot \\ e_1^T H^k X(t_{N_k}^k) \end{pmatrix} + \ldots + c_m' \begin{pmatrix} e_m^T H^k X(t_1^k) \\ \cdot \\ e_m^T H^k X(t_{N_k}^k) \end{pmatrix} \right) \\
&= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \begin{pmatrix} c_1' \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^{kT} \right) X(c)(t_i^k) - Z_i^k \right] e_1^T H^k X(t_i^k) + \ldots \\ + c_m' \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^{kT} \right) X(c)(t_i^k) - Z_i^k \right] e_m^T H^k X(t_i^k) \end{pmatrix} \\
&= c'^T \begin{pmatrix} \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^{kT} \right) X(c)(t_i^k) - Z_i^k \right] e_1^T H^k X(t_i^k) \\ \cdot \\ \cdot \\ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^{kT} \right) X(c)(t_i^k) - Z_i^k \right] e_m^T H^k X(t_i^k) \end{pmatrix}
\end{aligned}
$$

(3.6)

To calculate (*II*), we need to compute $(D\breve{X}(c)c')$.

From Equation (4), we have

$$
\begin{pmatrix} (DB_1(c)c')_{m\times m} & & & \\ -(D\widehat{B}_1(c)c')_{m\times m} & (DB_2(c)c')_{m\times m} & & \\ & \cdots & \cdots & \\ & & -(D\widehat{B}_{N-1}(c)c')_{m\times m} & (DB_N(c)c')_{m\times m} \end{pmatrix}_{mN\times mN} \breve{X}(c)
$$

$$
+ \begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & \widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN} (D\breve{X}(c)c')_{mN\times 1} = \begin{pmatrix} D\widehat{B}_0(c)c' X_0(c) \\ 0 \\ \cdots \\ 0 \end{pmatrix}_{mN\times 1}
$$

Then

$$\begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & -\widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN} (D\breve{X}(c)c')_{mN\times 1}$$

$$= \begin{pmatrix} D\widehat{B}_0(c)c'X_0(c) - DB_1(c)c'X_1(c) \\ D\widehat{B}_1(c)c'X_1(c) - DB_2(c)c'X_2(c) \\ \cdots \\ D\widehat{B}_{N-1}(c)c'X_{N-1}(c) - DB_N(c)c'X_N(c) \end{pmatrix}_{mN\times 1}$$

$$= dt \begin{pmatrix} (1-\theta)\xi_0 Adiag(c')X_0(c) - \theta\xi_1 Adiag(c')X_1(c) \\ \cdots \\ \cdots \\ (1-\theta)\xi_{N-1} Adiag(c')X_{N-1}(c) - \theta\xi_N Adiag(c')X_N(c) \end{pmatrix}_{mN\times 1}$$

$$= dt \begin{pmatrix} (1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c)) \\ (1-\theta)A\xi_1 diag(X_1) + \theta A\xi_2 diag(X_2(c)) \\ \cdots \\ (1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c)) \end{pmatrix} c'$$

i.e.,

$$\begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & -\widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix} (D\breve{X}(c)c')$$

$$= dt \begin{pmatrix} (1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c)) \\ (1-\theta)A\xi_1 diag(X_1) + \theta A\xi_2 diag(X_2(c)) \\ \cdots \\ (1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c)) \end{pmatrix} c'$$

(3.7)

Now let

$$p = \begin{pmatrix} p_1 \\ . \\ . \\ . \\ p_N \end{pmatrix}_{mN\times 1} \quad \text{where } p_j = \begin{pmatrix} p_{j,1} \\ . \\ . \\ . \\ p_{j,m} \end{pmatrix}_{m\times 1}, \text{ then } p^T = \begin{pmatrix} p_1^T & . & . & p_N^T \end{pmatrix}_{1\times mN} \text{ and}$$

$$p^T \begin{pmatrix} B_1(c) & & & \\ -\hat{B}_1(c) & B_2(c) & & \\ & . & . & \\ & & -\hat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN} D\breve{X}(c)c'$$

$$= dt\begin{pmatrix} (p_1^T)_{1\times m} & . & . & (p_N^T)_{1\times m} \end{pmatrix} \begin{pmatrix} \left((1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c))\right)_{m\times m} \\ . \\ . \\ \left((1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c))\right)_{m\times m} \end{pmatrix} c'$$

$$= dt\begin{Bmatrix} p_1^T\left((1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c))\right) + \ldots \\ + p_N^T\left((1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c))\right) \end{Bmatrix} c'$$

Set

$$p^T \begin{pmatrix} B_1(c) & & & \\ -\hat{B}_1(c) & B_2(c) & & \\ & \ldots & \ldots & \\ & & -\hat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN}$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c)P^k\breve{X}(c) - \breve{Z}^k]^T \Phi^k(c)(P^k)_{mN_k\times mN}$$

or take the transpose of both sides to give

$$\begin{pmatrix} B_1(c)^T & -\hat{B}_1(c)^T & & \\ & B_2(c)^T & -\hat{B}_2(c)^T & \\ & & \ldots & -\hat{B}_{N-1}(c)^T \\ & & & B_N(c)^T \end{pmatrix} p(c) = \sum_{k=1}^{s} \frac{1}{\sigma_k^2} P^{kT}\Phi^k(c)^T[\Phi^K(c)P^k\breve{X}(c) - \breve{Z}^k]$$

Solving the above equation gives the vector $p(c)$. Then

$$(II) = c'^{T} \begin{Bmatrix} \left((1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c))\right)^T p_1 \\ + \ldots + \left((1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c))\right)^T p_N \end{Bmatrix} dt \tag{3.8}$$

Plugging (3.6) and (3.8) into (3.4), we have:

$$\begin{aligned} DJ(c) = & \begin{pmatrix} \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^k\right) X(c)(t_i^k) - Z_i^k\right] e_1{}^T H^k X(t_i^k) \\ \cdots \\ \cdots \\ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} \left[\left(c^T H^k + \varphi^k\right) X(c)(t_i^k) - Z_i^k\right] e_m{}^T H^k X(t_i^k) \end{pmatrix}_{m\times 1} \\ & + dt \begin{Bmatrix} \left((1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c))\right)^T p_1 + \ldots \\ + \left((1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c))\right)^T p_N \end{Bmatrix}_{m\times 1} \\ & + \lambda G(c - c^o) \end{aligned} \tag{3.9}$$

3.2 Hessian matrix of the cost function $J(c)$

Starting From

$$\begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \ldots & \ldots & \\ & & -\widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix} (D\breve{X}(c)c')$$

$$= dt \begin{pmatrix} (1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c)) \\ (1-\theta)A\xi_1 diag(X_1) + \theta A\xi_2 diag(X_2(c)) \\ \ldots\ldots \\ (1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c)) \end{pmatrix} c'$$

we have

$$\begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \ddots & \ddots & \\ & & -\widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix}_{mN\times mN} \begin{pmatrix} [DX_1(c)c']_{m\times 1} \\ \vdots \\ \vdots \\ [DX_N(c)c']_{m\times 1} \end{pmatrix}_{mN\times 1}$$

$$= dt \begin{pmatrix} (1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c)) \\ (1-\theta)A\xi_1 diag(X_1) + \theta A\xi_2 diag(X_2(c)) \\ \vdots \\ (1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c)) \end{pmatrix} c'$$

i.e.,

$$B_1(c)[DX_1(c)c']_{m\times 1} = dt[(1-\theta)A\xi_0 diag(X_0) + \theta A\xi_1 diag(X_1(c))]c'$$

$$-\widehat{B}_1(c)[DX_1(c)c']_{m\times 1} + B_2(c)[DX_2(c)c']_{m\times 1} = dt[(1-\theta)A\xi_1 diag(X_1) + \theta A\xi_2 diag(X_2(c))]c'$$

$$\vdots$$

$$-\widehat{B}_{N-1}(c)[DX_{N-1}(c)c']_{m\times 1} + B_N(c)[DX_N(c)c']_{m\times 1} = dt[(1-\theta)A\xi_{N-1} diag(X_{N-1}) + \theta A\xi_N diag(X_N(c))]c'$$

Solving the above equation recursively gives:

$$(D\breve{X}(c)c')_{mN\times 1} = \eta(c)_{mN\times m}c' \tag{3.10}$$

Since $(I) = \sum_{k=1}^{s} \frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c) - \breve{Z}^k]^T(D\Phi^k(c)c')P^k\breve{X}(c)$

$$(II) = \sum_{k=1}^{s} \frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c) - \breve{Z}^k]^T\Phi^k(c)P^k(D\breve{X}(c)c')$$

we have

$$D^2J(c,c'') = D(I)c''+D(II)c''+D(III)c''$$

$$= \underbrace{\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[D\Phi^k(c)c''P^k\breve{X}(c)]^T(D\Phi^k(c)c')P^k\breve{X}(c)+}_{D^2J_1(c,c'')}$$

$$+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^kD\breve{X}(c)c'']^T(D\Phi^k(c)c')P^k\breve{X}(c)$$

$$\underbrace{+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[D\Phi^k(c)c''P^k\breve{X}(c)]^T\Phi^k(c)P^k(D\breve{X}(c)c')}_{D^2J_2(c,c'')}$$

$$+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c)-\breve{Z}^k]^T(D\Phi^k(c)c')P^kD\breve{X}(c)c''$$

$$\underbrace{+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c)-\breve{Z}^k]^TD\Phi^k(c)c''P^k(D\breve{X}(c)c')}_{D^2J_3(c,c'')}$$

$$\underbrace{+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^kD\breve{X}(c)c'']^T\Phi^k(c)P^k(D\breve{X}(c)c')_{N\times 1}}_{D^2J_4(c,c'')}$$

$$\underbrace{+\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c)-\breve{Z}^k]^T\Phi^k(c)P^k(D^2\breve{X}(c)(c',c''))_{N\times 1}}_{D^2J_5(c,c'')}$$

$$+\lambda G(c',c'')$$

$$= D^2J_1(c,c'')+D^2J_2(c,c'')+D^2J_3(c,c'')+D^2J_4(c,c'')+D^2J_5(c,c'')+\lambda G(c',c'')$$

(3.11)

Now

$$D^2J_1(c,c'') = \sum_{k=1}^{s}\frac{1}{\sigma_k^2}[D\Phi^k(c)c''P^k\breve{X}(c)]^T(D\Phi^k(c)c')P^k\breve{X}(c)$$

$$= \sum_{k=1}^{s}\frac{1}{\sigma_k^2}\breve{X}(c)^T{}_{1\times mN}\,P^{kT}{}_{mN\times mN_k}[D\Phi^k(c)c'']^T{}_{mN_k\times N_k}[D\Phi^k(c)c']_{N_k\times mN_k}P^k{}_{mN_k\times mN}\breve{X}(c)_{mN\times 1}$$

From the above

$$(D\Phi_k(c)c')_{(N_k\times mN_k)} = c_1'\begin{pmatrix} e_1{}^T H^k & & \\ & \ddots & \\ & & e_1{}^T H^k \end{pmatrix}_{(N_k\times mN_k)} + \ldots + c_m'\begin{pmatrix} e_m{}^T H^k & & \\ & \ddots & \\ & & e_m{}^T H^k \end{pmatrix}$$

Then

$$[D\Phi^k(c)c'']^T{}_{mN_k\times N_k}[D\Phi^k(c)c']_{N_k\times mN_k}$$

$$= \left(c_1''\begin{pmatrix} H^k e_1 & & \\ & \ddots & \\ & & H^k e_1 \end{pmatrix}_{(mN_k\times N_k)} + \ldots + c_m''\begin{pmatrix} H^k e_m & & \\ & \ddots & \\ & & H^k e_m \end{pmatrix}_{(mN_k\times N_k)} \right)$$

$$\times \left(c_1'\begin{pmatrix} e_1{}^T H^k & & \\ & \ddots & \\ & & e_1{}^T H^k \end{pmatrix}_{(N_k\times mN_k)} + \ldots + c_m'\begin{pmatrix} e_m{}^T H^k & & \\ & \ddots & \\ & & e_m{}^T H^k \end{pmatrix}_{(N_k\times mN_k)} \right)$$

$$= c_1'c_1''\begin{pmatrix} H^k e_1 e_1{}^T H^k & & \\ & \ddots & \\ & & H^k e_1 e_1{}^T H^k \end{pmatrix}_{(mN_k\times mN_k)} + \ldots$$

$$+ c_i'c_j''\begin{pmatrix} H^k e_i e_j{}^T H^k & & \\ & \ddots & \\ & & H^k e_i e_j{}^T H^k \end{pmatrix}_{(mN_k\times mN_k)} + \ldots$$

$$+ c_m'c_m''\begin{pmatrix} H^k e_m e_m{}^T H^k & & \\ & \ddots & \\ & & H^k e_m e_m{}^T H^k \end{pmatrix}_{(mN_k\times mN_k)}$$

Therefore:

$$D^2 J_1(c', c'')$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \breve{X}(c)^T \, {P^k}^T{}_{mN \times mN_k} \begin{pmatrix} c_1' c_1'' \begin{pmatrix} H^k e_1 e_1^T H^k & & \\ & \ddots & \\ & & H^k e_1 e_1^T H^k \end{pmatrix}_{(mN_k \times mN_k)} + \\ \ldots + c_m' c_m'' \begin{pmatrix} H^k e_m e_m^T H^k & & \\ & \ddots & \\ & & H^k e_m e_m^T H^k \end{pmatrix} \end{pmatrix} P^k \breve{X}(c)$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \breve{X}(c)^T \, {P^k}^T \left(c_1' c_1'' \begin{pmatrix} H^k e_1 e_1^T H^k X(t_1^k) \\ \vdots \\ H^k e_1 e_1^T H^k X(t_{N_k}^k) \end{pmatrix} + \ldots + c_m' c_m'' \begin{pmatrix} H^k e_m e_m^T H^k X(t_1^k) \\ \vdots \\ H^k e_m e_m^T H^k X(t_{N_k}^k) \end{pmatrix} \right)$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \left(c_1' c_1'' \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_1^T H^k X(t_i^k) + .. + c_m' c_m'' \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_m e_m^T H^k X(t_i^k) \right)$$

$$= c'^T \begin{pmatrix} \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_1^T H^k X(t_i^k) & .. & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_m^T H^k X(t_i^k) \\ \ldots & .. & \ldots \\ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_m^T H^k X(t_i^k) & .. & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_m e_m^T H^k X(t_i^k) \end{pmatrix} c''$$

$$= c'^T M_1 c''$$

where $M_1 = \begin{pmatrix} \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_1^T H^k X(t_i^k) & .. & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_m^T H^k X(t_i^k) \\ \ldots & .. & \ldots \\ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_1 e_m^T H^k X(t_i^k) & .. & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \sum_{i=1}^{N_k} X(t_i^k)^T H^k e_m e_m^T H^k X(t_i^k) \end{pmatrix}_{m \times m}$$

is a symmetric matrix.

Next:

$$
\begin{aligned}
D^2 J_2(c',c'') &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k D\breve{X}(c) c'']^T (D\Phi^k(c)c') P^k \breve{X}(c) \\
&\quad + \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [D\Phi^k(c)c'' P^k \breve{X}(c)]^T \Phi^k(c) P^k (D\breve{X}(c)c') \\
&= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [D\breve{X}(c)c'']^T {P^k}^T \Phi^k(c)^T (D\Phi^k(c)c') P^k \breve{X}(c) \\
&\quad + \sum_{k=1}^{s} \frac{1}{\sigma_k^2} \breve{X}(c)^T {P^k}^T [D\Phi^k(c)c'']^T \Phi^k(c) P^k (D\breve{X}(c)c') \\
&= \Delta_2 + {\Delta_2}^T
\end{aligned}
$$

and

$$
\begin{aligned}
\Delta_2 &= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k (\eta(c)_{mN\times m} c'')]^T [D\Phi^k(c)c'] P^k \breve{X}(c) \\
&= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\eta(c)_{mN\times m} c'']^T {P^k}^T \Phi^k(c)^T [D\Phi^k(c)c'] P^k \breve{X}(c) \\
&= \sum_{k=1}^{s} \left(\frac{1}{\sigma_k^2} c''^T \eta(c)^T{}_{m\times mN} {P^k}^T \Phi^k(c)^T \left(c_1' \begin{pmatrix} {e_1}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_1}^T H^k X(t_{N_k}^k) \end{pmatrix}_{N_k\times 1} + \ldots + c_m' \begin{pmatrix} {e_m}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_m}^T H^k X(t_{N_k}^k) \end{pmatrix} \right) \right) \\
&= c''^T \left(\sum_{k=1}^{s} \frac{1}{\sigma_k^2} \eta(c)^T {P^k}^T{}_{mN\times mN_k} \Phi^k(c)^T{}_{mN_k\times N_k} \left(\begin{pmatrix} {e_1}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_1}^T H^k X(t_{N_k}^k) \end{pmatrix}_{N_k\times 1} \quad \cdots \quad \begin{pmatrix} {e_m}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_m}^T H^k X(t_{N_k}^k) \end{pmatrix} \right) \right) c'
\end{aligned}
$$

Set

$$
{M_2}^T = \left(\sum_{k=1}^{s} \frac{1}{\sigma_k^2} \eta(c)^T {P^k}^T{}_{mN\times mN_k} \Phi^k(c)^T{}_{mN_k\times N_k} \left(\begin{pmatrix} {e_1}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_1}^T H^k X(t_{N_k}^k) \end{pmatrix}_{N_k\times 1} \quad \cdots \quad \begin{pmatrix} {e_m}^T H^k X(t_1^k) \\ \cdots \\ \cdots \\ {e_m}^T H^k X(t_{N_k}^k) \end{pmatrix} \right) \right)
$$

Then

$$D^2 J_2(c', c'') = \Delta_2 + \Delta_2{}^T = c'^T (M_2 + M_2{}^T) c''$$

Next:

$$D^2 J_3(c, c'')$$
$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T (D\Phi^k(c)c')_{N_k \times mN_k} P^k{}_{mN_k \times mN} (D\breve{X}(c)c'')$$
$$+ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T (D\Phi^k(c)c'') P^k (D\breve{X}(c)c')$$
$$= \Delta_3 + \Delta_3{}^T$$

$$(\Delta_3) = \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T (D\Phi^k(c)c') P^k D\breve{X}(c)c''$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T{}_{1 \times N_k} [D\Phi^k(c)c']_{N_k \times mN_k} P^k{}_{mN_k \times mN} [\eta(c)_{mN \times m} c'']_{mN \times 1}$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T{}_{1 \times N_k} \begin{pmatrix} c_1' \begin{pmatrix} e_1{}^T H^k & & \\ & \cdots & \\ & & e_1{}^T H^k \end{pmatrix} + \cdots \\ + c_m' \begin{pmatrix} e_m{}^T H^k & & \\ & \cdots & \\ & & e_m{}^T H^k \end{pmatrix}_{(N_k \times mN_k)} \end{pmatrix} P^k [\eta(c)c'']_{mN \times 1}$$

$$= \sum_{i=1}^{m} c_i' \left(\sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T{}_{1 \times N_k} \begin{pmatrix} e_i{}^T H^k & & \\ & \cdot & \\ & & e_i{}^T H^k \end{pmatrix}_{(N_k \times mN_k)} P^k{}_{mN_k \times mN} \right) \eta(c)c''$$

$$= \sum_{i=1}^{m} c_i' \left(\left(\sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_1^k) - Z_1^k] e_i{}^T H^k \quad \cdots \quad \cdots \quad \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_{N_k}^k) - Z_{N_k}^k] e_i{}^T H^k \right) P^k \right) \eta(c)c''$$

$$= c'^T \left(\begin{pmatrix} \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_1^k) - Z_1^k] e_1{}^T H^k & \cdots & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_{N_k}^k) - Z_{N_k}^k] e_1{}^T H^k \\ \cdots & \cdots & \cdots \\ \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_1^k) - Z_1^k] e_m{}^T H^k & \cdots & \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [c^T H^K X(t_{N_k}^k) - Z_{N_k}^k] e_m{}^T H^k \end{pmatrix} P^k \eta(c) \right) c''$$

$$= c'^T M_3 c''$$

$$D^2 J_3(c',c'') = \Delta_3 + \Delta_3{}^T = c'^T (M_3 + M_3{}^T) c''$$

Now

$$D^2 J_4(c',c'') = \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k D\breve{X}(c) c'']^T \Phi^k(c) P^k (D\breve{X}(c) c')_{mN\times 1}$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \eta(c)_{mN\times m} c'']^T \Phi^k(c) P^k (\eta(c)_{mN\times m} c')_{mN\times 1}$$

$$= \sum_{k=1}^{s} \frac{1}{\sigma_k^2} c''^T \eta(c)_{m\times mN}{}^T P^{kT} \Phi^k(c)^T \Phi^k(c) P^k \eta(c)_{mN\times m} c'$$

$$= c'^T \left(\sum_{k=1}^{s} \frac{1}{\sigma_k^2} \eta(c)_{m\times mN}{}^T P^{kT} \Phi^k(c)^T \Phi^k(c) P^k \eta(c)_{mN\times m} \right) c''$$

$$= c'^T M_4 c''$$

Now we calculate:

$$D^2 J_5(c',c'') = \sum_{k=1}^{s} \frac{1}{\sigma_k^2} [\Phi^k(c) P^k \breve{X}(c) - \breve{Z}^k]^T \Phi^k(c) P^k (D^2 \breve{X}(c)(c',c''))_{mN\times 1}$$

From

$$\begin{pmatrix} B_1(c) & & & \\ -\widehat{B}_1(c) & B_2(c) & & \\ & \cdots & \cdots & \\ & & -\widehat{B}_{N-1}(c) & B_N(c) \end{pmatrix} \breve{X}(c) = \begin{pmatrix} \widehat{B}_0(c) X_0(c) \\ 0 \\ \cdots \\ 0 \end{pmatrix} + \breve{f}$$

$$(D\widetilde{B}(c)c')_{mN\times mN} \breve{X}(c) + \widetilde{B}(c)_{mN\times mN} D\breve{X}(c)c' = D \begin{pmatrix} \widehat{B}_0(c) X_0(c) \\ 0 \\ .... \\ 0 \end{pmatrix} c'$$

$$(D\widetilde{B}(c)c')_{mN\times mN} D\breve{X}(c)c'' + (D\widetilde{B}(c)c'')_{mN\times mN} D\breve{X}(c)c' + \widetilde{B}(c)_{mN\times mN} D^2 \breve{X}(c)(c',c'') = 0$$

$$\widetilde{B}(c)_{mN\times mN} (D^2 \breve{X}(c)(c',c''))_{mN\times 1} = -(D\widetilde{B}(c)c')_{mN\times mN} D\breve{X}(c)c'' - (D\widetilde{B}(c)c'')_{mN\times mN} D\breve{X}(c)c'$$

$$\widetilde{B}(c)_{mN\times mN}(D^2\breve{X}(c)(c',c''))_{mN\times 1}$$

$$=-\begin{pmatrix} (DB_1(c)c')_{m\times m} & & & \\ -(D\hat{B}_1(c)c')_{m\times m} & (DB_2(c)c')_{m\times m} & & \\ & \cdots & \cdots & \\ & & -(D\hat{B}_{N-1}(c)c')_{m\times m} & (DB_N(c)c')_{m\times m} \end{pmatrix}_{mN\times mN}\begin{pmatrix} D\breve{X}_1(c)c'' \\ \cdots \\ \cdots \\ \cdots \\ D\breve{X}_N(c)c'' \end{pmatrix}$$

$$-\begin{pmatrix} (DB_1(c)c'')_{m\times m} & & & \\ -(D\hat{B}_1(c)c'')_{m\times m} & (DB_2(c)c'')_{m\times m} & & \\ & \cdots & \cdots & \\ & & -(D\hat{B}_{N-1}(c)c'')_{m\times m} & (DB_N(c)c'')_{m\times m} \end{pmatrix}_{mN\times mN}\begin{pmatrix} D\breve{X}_1(c)c' \\ \cdots \\ \cdots \\ \cdots \\ D\breve{X}_N(c)c' \end{pmatrix}$$

$$=\begin{pmatrix} -(DB_1(c)c')D\breve{X}_1(c)c''-(DB_1(c)c'')D\breve{X}_1(c)c' \\ (D\hat{B}_1(c)c')D\breve{X}_1(c)c''-(DB_2(c)c')D\breve{X}_2(c)c''+(D\hat{B}_1(c)c'')D\breve{X}_1(c)c'-(DB_2(c)c'')D\breve{X}_2(c)c' \\ \cdots \\ (D\hat{B}_{N-1}(c)c')D\breve{X}_{N-1}(c)c''-(DB_N(c)c')D\breve{X}_N(c)c''+(D\hat{B}_{N-1}(c)c'')D\breve{X}_{N-1}(c)c'-(DB_N(c)c'')D\breve{X}_N(c)c' \end{pmatrix}$$

$$=dt\begin{pmatrix} \theta\xi_1 Adiag(c')\eta_1(c)c''+\theta\xi_1 Adiag(c'')\eta_1(c)c' \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \left((1-\theta)\xi_{N-1}Adiag(c')\eta_{N-1}(c)+\theta\xi_N Adiag(c')\eta_N(c)\right)c''+ \\ \left((1-\theta)\xi_{N-1}Adiag(c'')\eta_{N-1}(c)+\theta\xi_N Adiag(c'')\eta_N(c)\right)c' \end{pmatrix}$$

Let

$$p^T\widetilde{B}(c)_{mN\times mN}=\sum_{k=1}^{s}\frac{1}{\sigma_k^2}[\Phi^k(c)P^k\breve{X}(c)-\breve{Z}^k]^T\Phi^k(c)P^k$$

then

$$
\begin{aligned}
& p^T \widetilde{B}(c)_{mN\times mN}\left(D^2 \breve{X}(c)(c',c'')\right)_{mN\times 1} \\
&= p^T dt \begin{pmatrix} \theta\xi_1 Adiag(c')\eta_1(c)c''+\theta\xi_1 Adiag(c'')\eta_1(c)c' \\ \cdot \\ \cdot \\ \left((1-\theta)\xi_{N-1} Adiag(c')\eta_{N-1}(c)+\theta\xi_N Adiag(c')\eta_N(c)\right)c''+ \\ \left((1-\theta)\xi_{N-1} Adiag(c'')\eta_{N-1}(c)+\theta\xi_N Adiag(c'')\eta_N(c)\right)c' \end{pmatrix} \\
&= dt\times \\
&\left\{ \begin{matrix} p_1{}^T\left(\theta\xi_1 Adiag(c')\eta_1(c)c''+\theta\xi_1 Adiag(c'')\eta_1(c)c'\right)+\ldots+ p_N{}^T \times \\ \times\left(\begin{matrix} \left((1-\theta)\xi_{N-1} Adiag(c')\eta_{N-1}(c)+\theta\xi_N Adiag(c')\eta_N(c)\right)c''+ \\ +\left((1-\theta)\xi_{N-1} Adiag(c'')\eta_{N-1}(c)+\theta\xi_N Adiag(c'')\eta_N(c)\right)c' \end{matrix} \right) \end{matrix} \right\}
\end{aligned}
$$

Now

$$
\begin{aligned}
& p_1{}^T\left(\theta\xi_1 Adiag(c')\eta_1(c)c''+\theta\xi_1 Adiag(c'')\eta_1(c)c'\right) \\
&= \sum_{i-1}^{m} c_i' \, p_1{}^T\theta\xi_1 AE_i\eta_1(c)c''+c_i'' \, p_1{}^T\theta\xi_1 AE_i\eta_1(c)c' \\
&= c'^T \begin{pmatrix} p_1{}^T\theta\xi_1 AE_1\eta_1(c) \\ \cdot \\ \cdot \\ p_1{}^T\theta\xi_1 AE_m\eta_1(c) \end{pmatrix}_{m\times m} c''+c''^T \begin{pmatrix} p_1{}^T\theta\xi_1 AE_1\eta_1(c) \\ \cdot \\ \cdot \\ p_1{}^T\theta\xi_1 AE_m\eta_1(c) \end{pmatrix}_{m\times m} c' \\
&= c'^T \, M^{(1)}c''+c'^T \, M^{(1)^T}c'' = c'^T \, (M^{(1)}+M^{(1)^T})c''
\end{aligned}
$$

where

$$
M^{(1)} = \begin{pmatrix} p_1{}^T\theta\xi_1 AE_1\eta_1(c) \\ \cdot \\ \cdot \\ p_1{}^T\theta\xi_1 AE_m\eta_1(c) \end{pmatrix}_{m\times m}
$$

and E_i is the matrix with the *i*-th diagonal element being 1 and the remaining elements being zero

Similarly

$$
\begin{aligned}
& p_N{}^T \begin{pmatrix} \left((1-\theta)\xi_{N-1} Adiag(c')\eta_{N-1}(c) + \theta\xi_N Adiag(c')\eta_N(c)\right)c'' \\ + \left((1-\theta)\xi_{N-1} Adiag(c'')\eta_{N-1}(c) + \theta\xi_N Adiag(c'')\eta_N(c)\right)c' \end{pmatrix} \\
& = \sum_{i=1}^{m} c_i' \, p_N{}^T \{(1-\theta)\xi_{N-1} AE_i \eta_{N-1}(c) + \theta\xi_N AE_i \eta_N(c)\}c'' \\
& + \sum_{i=1}^{m} c_i'' \, p_N{}^T \{(1-\theta)\xi_{N-1} AE_i \eta_{N-1}(c) + \theta\xi_N AE_i \eta_N(c)\}c'' \\
& = c'^T \begin{pmatrix} p_N{}^T \{(1-\theta)\xi_{N-1} AE_1 \eta_{N-1}(c) + \theta\xi_N AE_1 \eta_N(c)\} \\ \cdot \\ \cdot \\ p_N{}^T \{(1-\theta)\xi_{N-1} AE_N \eta_{N-1}(c) + \theta\xi_N AE_N \eta_N(c)\} \end{pmatrix}_{m\times m} c'' \\
& + c''^T \begin{pmatrix} p_N{}^T \{(1-\theta)\xi_{N-1} AE_1 \eta_{N-1}(c) + \theta\xi_N AE_1 \eta_N(c)\} \\ \cdot \\ \cdot \\ p_N{}^T \{(1-\theta)\xi_{N-1} AE_N \eta_{N-1}(c) + \theta\xi_N AE_N \eta_N(c)\} \end{pmatrix}_{m\times m} c' \\
& = c'^T \, M^{(N)} c'' + c''^T \, M^{(N)^T} c'' = c'^T \, (M^{(N)} + M^{(N)^T}) c''
\end{aligned}
$$

where

$$
M^{(N)} = \begin{pmatrix} p_N{}^T \{(1-\theta)\xi_{N-1} AE_1 \eta_{N-1}(c) + \theta\xi_N AE_1 \eta_N(c)\} \\ \cdot \\ \cdot \\ p_N{}^T \{(1-\theta)\xi_{N-1} AE_N \eta_{N-1}(c) + \theta\xi_N AE_N \eta_N(c)\} \end{pmatrix}_{m\times m}
$$

We then have:

$$
\begin{aligned}
& D^2 J_5(c', c'') = dt\{p_1{}^T \left(\theta\xi_1 Adiag(c')\eta_1(c)c'' + \theta\xi_1 Adiag(c'')\eta_1(c)c'\right) + \ldots \\
& + p_N{}^T \begin{pmatrix} \left((1-\theta)\xi_{N-1} Adiag(c')\eta_{N-1}(c) + \theta\xi_N Adiag(c')\eta_N(c)\right)c'' \\ + \left((1-\theta)\xi_{N-1} Adiag(c'')\eta_{N-1}(c) + \theta\xi_N Adiag(c'')\eta_N(c)\right)c' \end{pmatrix} \\
& = dt\left(c'^T \left(\sum_{i=1}^{N} \left(M^{(i)} + M^{(i)^T} \right) \right) c'' \right)
\end{aligned}
$$

Since each component in (3.11) has been calculated, summing them together we will have $D^2 J(c, c'')$.

3.3 Minimization algorithms

Now that the Jacobian and the Hessian of the cost function $J(c)$ have been developed, we can introduce various deterministic optimization techniques. We refer to Press et al. (1992) for a complete discussion of various numerical optimization techniques that rely on the first derivative and the second derivative information. In the next, for numerical testing purposes we introduce two algorithms: the constrained steepest descent algorithm and the constrained conjugate gradient algorithm. We propose a feedback mechanism that utilizes the performance of the algorithm at the current stage to ensure a faster convergence of the search process. These two algorithms only involve the first derivative of $J(c)$. We try to avoid using the algorithms that rely on the second derivatives (Hessian) on the consideration that in the case of ill-constrained inverse problems, the second derivative of the cost function with respect to some of the parameters may be nearly zero if no regularization scheme is used, i.e., the Hessian matrix may be close to singular, as is seen in the inverse problem of the seven-compartment model. In the following Ω is an admissible set consisting of the Cartesian product of the closed intervals.

3.3.1 Constrained steepest descent optimization method.

Define the absolute error between two consecutive search points as $e = | J(c) - J(c_1) |$. The proposed algorithm is given in the following:

Step 1. Start from $c = c^{priot}$ or any point that is close to c^{priot}. Calculate $DJ(c)$ and

$$u = \frac{DJ(c)}{\| DJ(c) \|} .$$

Step 2. Take $k = n_1,..,n_2$, for each k calculate $J\left(c - \frac{g(e)}{m^k}u\right)$ where $n_2 > n_1 > 0, m > 0$ are integers and $g(e)$ is an increasing function of e taking positive values. Take the k that gives J the largest decrease and then let $c^1 = c - \frac{g(e)}{m^k}u$. If any component in vector c hits its upper bound U, then generate a random number $\rho_1 \in (0.9, \ 1)$ and replace the component with $\rho_1 U$; if it hits the lower bound L, then generate a random number $\rho_2 \in (1, \ 1.1)$, and replace it with $\rho_2 L$. This ensures that the component is in the interior near the upper or lower bound.

Step 3. Calculate $e = | J(c) - J(c_1) |$ **,** terminate the algorithm if it is less than a threshold number δ. Otherwise calculate $g(e)$ and go to step 4.

Step 4. Set $c = c^1$, calculate $u = \frac{DJ(c)}{\| DJ(c) \|}$ go to step 2.

Note that when minimizing a simple function, one can calculate the optimal descent step along the gradient direction by calculating $\alpha > 0$ such that $J(c - \alpha \nabla c)$ is minimized. Since this is impossible in our case, we use the Armijo rule in step 2 as was done in White (2002), i.e., several probing steps are tried and the integer k that gives the largest cost function decrease is used in the descent step size. To make the descent fast during the initial and middle stage of searching, we introduce the function $g(e)$ to control the descent steps based on the current performance indicated by e. It is an increasing function of e with a minimum lager than a positive constant. $g(e)$ is bounded below to ensure the updating step sizes are not too small at the end of the search (Figure 3.1).

3.3.2 Constrained conjugate gradient optimization method.

The proposed method is similar as the above except that the search direction is now updated to a conjugate direction that is more efficient for descending. The update formula can be either taken as the Fletcher-Reeves formula or the Polak-Ribiere formula (Press 1992), as is shown in the following step 5. The algorithm is thus:

Step 1. Start from $c = c^{priot}$ or any point that is close to c^{priot}. Calculate $DJ(c)$ and $u^0 = \frac{DJ(c)}{\| DJ(c) \|}$.

Step 2. Same as the above Step 2.

Step 3. Same as the above step 3.

Step 4. Set $c = c^1$, calculate $u = \frac{DJ(c)}{\| DJ(c) \|}$.

Step 5. Let $\beta = < u, u > / < u^0, u^0 >$ (Fletcher-Reeves formula) or let $\beta = < u, u - u^0 > / < u, u >$ (Polak-Ribiere formula). Update u to $u + \beta u^0$. Go to step 2.

3.4 Numerical results

We applied the above algorithms to the carbon model described in Chapter 2.3. In this example, we assume that no prior information is given on the model parameters. Therefore three runs were conducted, each starting with a different initial point in the admissible set. To make the solution of Model (2.1) and correspondingly $J(c)$

continuously differentiable, the smoothed versions of the two time series $u(.)$, $\xi(.)$ were used to remove the random noise. The number of probing steps was set to be five with $n_1 = 1, n_2 = 5, m = 2$. The feedback function $g(e)$ was chosen to be a piecewise linear increasing function of e shown in Figure 3.1. The searches were terminated when $e < 10^{-4}$. The cost function value when stopping criterion was met was about 36. The results from the three runs are shown in Figure 3.2.

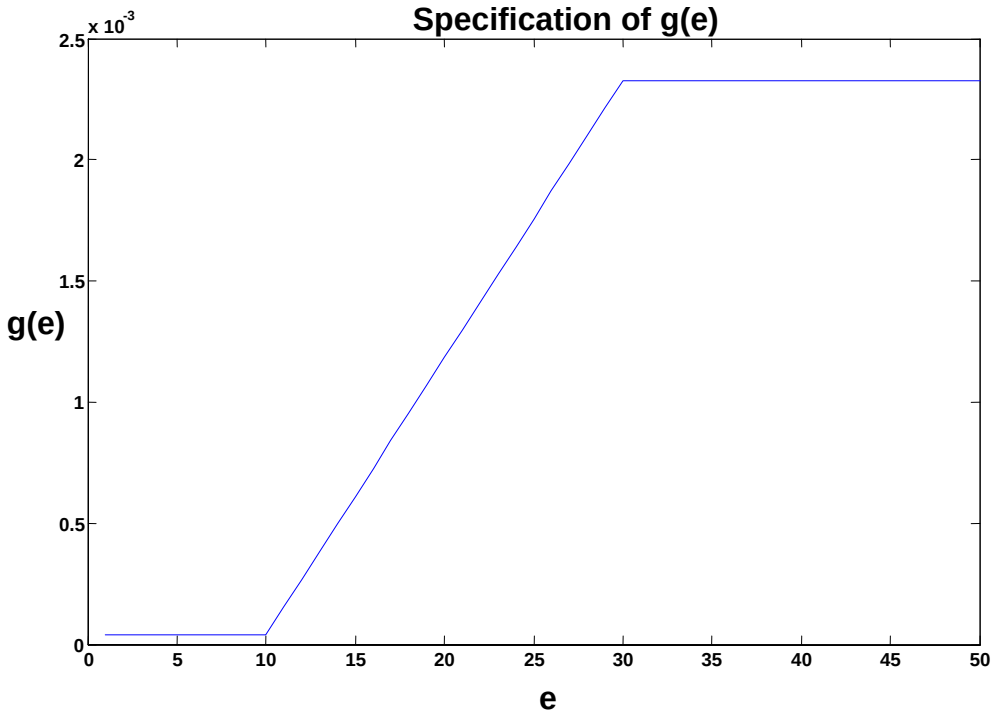

Figure 3.1 One example for specifying $g(e)$. This function was used as a feedback to control the descent step sizes. In general it is an increasing function of $e_k = | J(c_k) - J(c_{k-1}) |$. Other forms of feedback functions are also possible. Note however, the minimum value of $g(e)$ is a constant that is bigger than a positive number to prevent the step sizes from being too small.

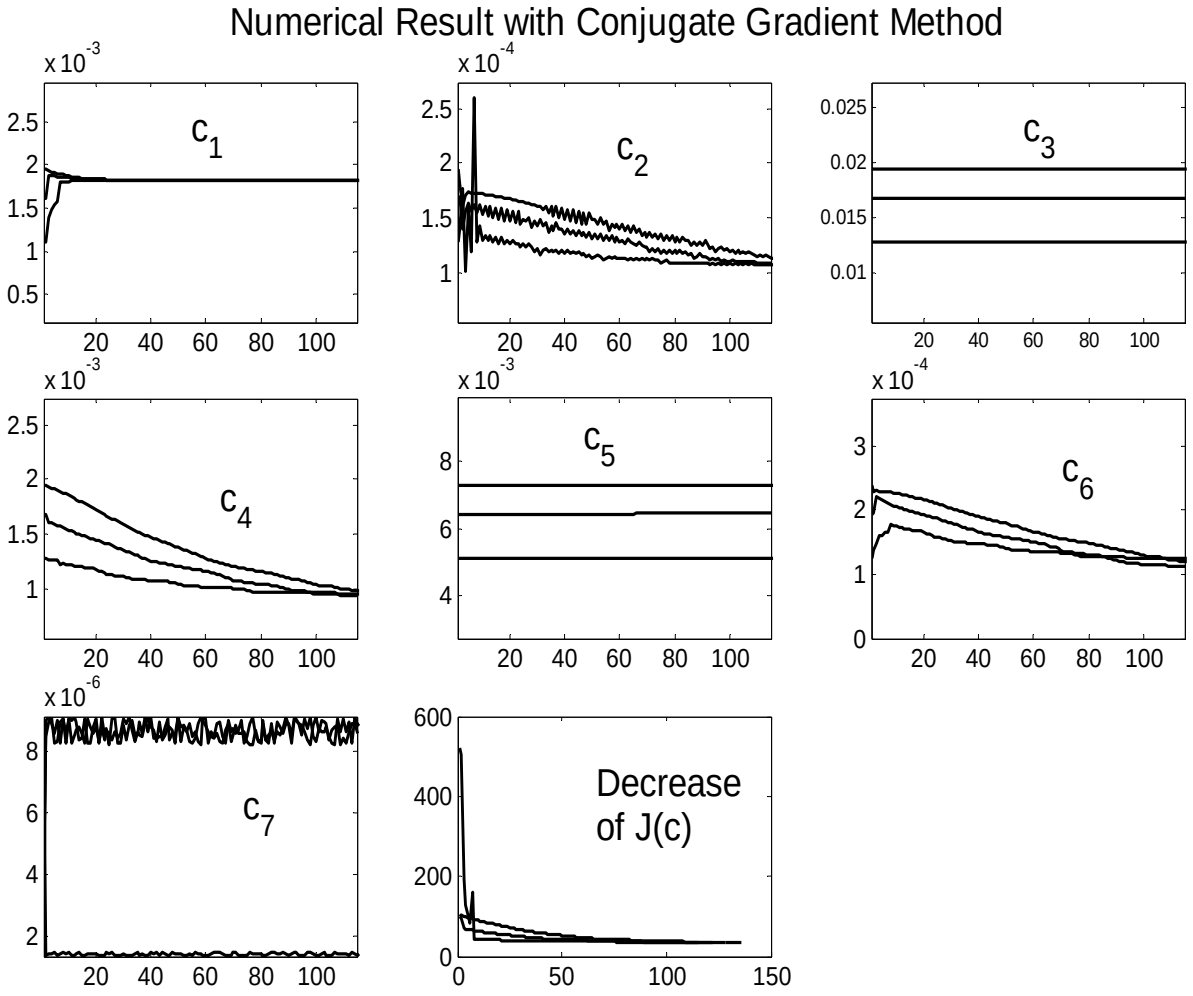

Figure 3.2. Trace plots of the search processes for optimal point. The figure shows the convergences or divergences of parameters and also the decrease of the cost functional $J(c)$. On each plot, the x-axis represents the number of descent steps. The y-axis for the plots of $c_1, \ldots, c_7$ are the physical ranges of the parameters.

From Figure 3.2 it can be observed that starting from different initial positions, the searching paths for each of c_1, c_2, c_4 and c_6 converge to the same point, the searching paths for c_3, c_5 remain constant, and the searching paths for c_7 either hit the upper bound or the lower bound. Thus the inverse problem seems only meaningful for c_1, c_2, c_4 and c_6 in that the data space, model structure and parameter space together give information to these four parameters. There are visible overshootings for parameter c_2 due to the use of the Armijo rule; however, the magnitude decreases as the search is near to the optimal as the gradient closes to zero. The cost function decreases rapidly in general at the initial and middle search stage since the function $g(e)$ gives relatively large probing steps. The observation that the three descending paths starting from different initial points for each of c_1, c_2, c_4, c_6 converges to the same point might suggest the cost function $J(c)$ is unimodal for c_1, c_2, c_4, c_6.

In general, the above analysis suggests that enough information can be given to c_1, c_2, c_4 and c_6 through inverse analysis, but not to c_3, c_5, c_7. These observations will be checked by other methods in the following sections.

Using one of the searching results we have the following estimation for the identifiable values:

$$c^o = \left(1.8238\times10^{-3},\quad 1.0933\times10^{-4},\quad \text{-},\quad 0.9862\times10^{-3},\quad \text{-},\quad 1.1471\times10^{-4},\quad \text{-}\right)$$

where "-" stands for non-available.

CHAPTER 4

STOCHASTIC OPTIMIZATION

The deterministic algorithms are suitable only when the cost function has one mode; otherwise a direct application of a gradient type method would result in finding a minimum that may be local and not the global one. In cases where there is no prior information about the optimal point or the behavior of the cost function on the parameter space is unknown, it is hard for deterministic approach to locate a global optimal. Another drawback of the deterministic approaches is that the functions in Model (2.1) must satisfy certain conditions (e.g., being Lipschitz continuous) to make $J(c)$ continuously differentiable. In the above numerical analysis, we used smoothed versions of $\xi(t)$ and $U(t)$ shown in Figure (2.2) and the noises and jumps are thus removed. But these disturbances are commonly seen in ecosystem modeling and should not be ignored. It is necessary to introduce other types of optimization techniques that are robust with

optimization under noise and do not need the gradient information. Stochastic optimization techniques are a natural choice.

Stochastic optimization techniques such as simulated annealing (SA), genetic algorithm (GA), neural network, downhill simplex method, etc., are powerful global optimization techniques. The theoretical foundations and applications of random search are partially documented in books by Goldberg (1989), Holland (1992), Haupt (1998), Vose (1999). Convergence of most random search procedures is not affected by the cost function, in particular its smoothness and multimodality. In a minimax sense, stochastic search is more powerful than deterministic search: it is nearly the best method in the worst possible situation (noise, discontinuity, multimodality) and the worst method in the best situation (smoothness, continuity, unimodality).

A detailed theoretical explanation of the general stochastic algorithms is beyond the scope of this thesis. Here we focus on the application of the algorithms to the Model (2.1) ~ (2.6). In the next we give two stochastic optimization algorithms and some general specifications. We then apply them to Model (2.1) ~ (2.6) in the context of the seven-pool compartmental model. We also compare the results with the result of the previous section.

4.1 GA Description

4.1.1 General description of genetic algorithm

GA was first invented by John Holland in 1975 and later popularized by his students and colleagues De Jong and Hollstien, etc. (Goldberg 1989). It is a global optimization method based on simulating nature's evolution and selection processes. The algorithm follows an over-population, selection, crossover and mutation type of procedure in order to select the fittest set of parameters (global optimal points). The structure of GA is the following:

GA algorithm

Step1. *(Initializing)* Randomly generate a population of n chromosomes in the admissible set.

Step 2. *(Fitness evaluation)* Evaluate the fitness specified by the cost functional $J(c)$ for each of the chromosomes in the population.

Step 3. *(Creating new population)* Generate each new population by the following steps:

- *(Selection)* Select two parent chromosomes from the current population according to their fitness.
- *(Crossover)* With pre-specified crossover probability cross over the parents to form new offspring.
- *(Mutation)* With a mutation probability mutate the set of offspring.
- *(Accepting)* Place new offspring in the new population.

Step 4. (*Replace)* Replace the previous population with the new population.

Step 5. *(Test and stop)* If the end condition is satisfied, terminate the program and return the best solution in current population; otherwise go to step 2.

The structure is shown in Figure 4.1.

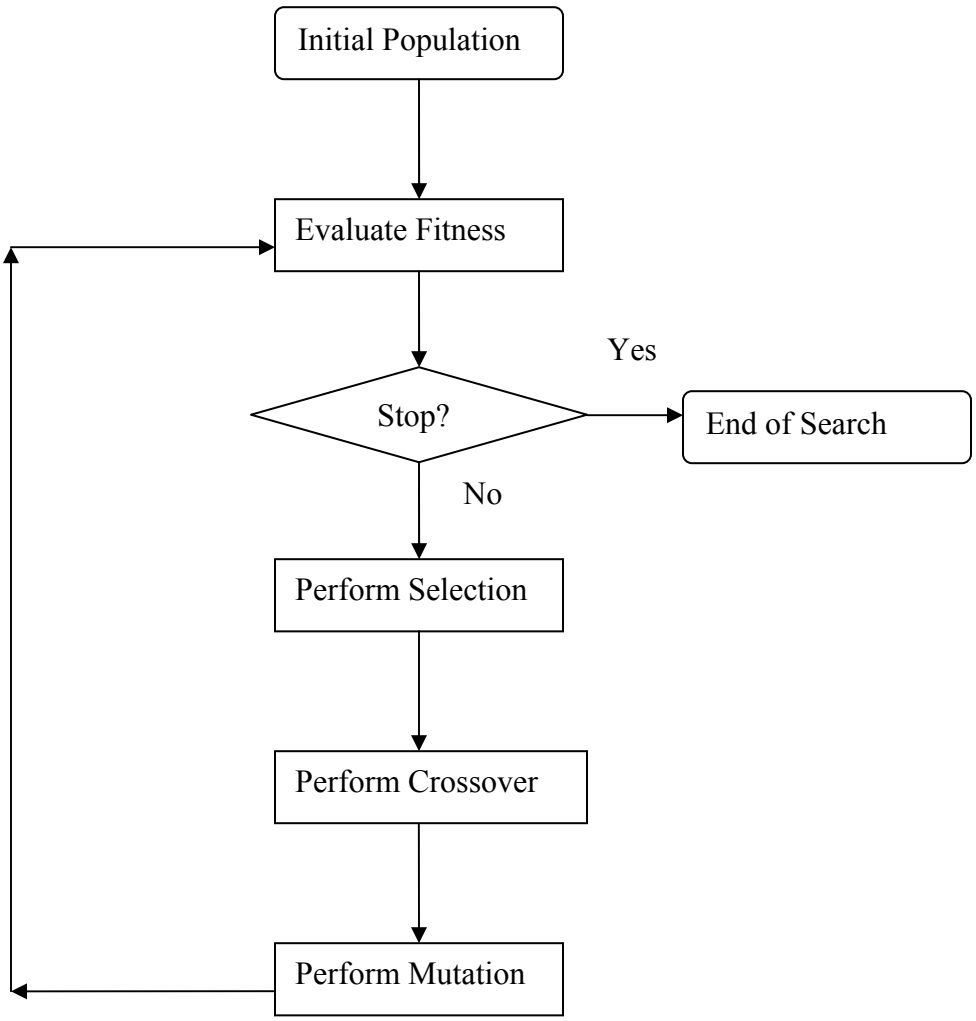

Figure 4.1: Flowchart of GA

4.1.2 Specifications of GA with respect to Model (2.1) ~ (2.6)

To implement GA with Model (2.1) ~ (2.6), some specifications must be given.

Parameter encoding. Depending on the problems at hand, there are various ways of encoding the parameter space such as the binary encoding, the permutation encoding, the direct value encoding, etc. (Haupt et al. 1998). For parameter estimation problems, the most commonly used encodings are either binary encoding or direct value encoding. Direct value encoding can be used in problems where complicated values such as real numbers are used. We use direct value encoding in this study since the parameters in Model (2.1) are real numbers. Thus each parameter vector is directly written in the form of $c=\left(c_1 \ . \ . \ c_m\right)$.

Crossover. Crossover operates on selected genes from parent chromosomes and creates new offspring. The simplest way is to randomly choose some crossover point and copy everything before this point from one parent and then copy everything after the crossover point from the other parent.

Chromosome 1	$\left(c_1^{(1)} \quad c_2^{(1)} \quad c_3^{(1)} \mid \ c_4^{(1)} \quad . \quad . \quad c_m^{(1)}\right)$
Chromosome 2	$\left(c_1^{(2)} \quad c_2^{(2)} \quad c_3^{(2)} \mid \ c_4^{(2)} \quad . \quad . \quad c_m^{(2)}\right)$
Offspring 1	$\left(c_1^{(2)} \quad c_2^{(2)} \quad c_3^{(2)} \mid \ c_4^{(1)} \quad . \quad . \quad c_m^{(1)}\right)$
Offspring 2	$\left(c_1^{(1)} \quad c_2^{(1)} \quad c_3^{(1)} \mid \ c_4^{(2)} \quad . \quad . \quad c_m^{(2)}\right)$

Table 4.1. Crossing over by swapping in which the third "gene" of Chromosome 1 and Chromosome 2 are chosen and swapped to generate two offsprings.

In this study, we choose to take the convex linear combination (Haupt et al. 1998) of the "genes" since the admissible set in our study is a closed convex set made up of Cartesian product of closed intervals (see the set made up of intervals in (2.7)), and the convex linear combination will reproduce points in the set. For this purpose, *m* random numbers $\theta_i \in [0,1], i = 1,...,m$ uniformly distributed between [0, 1] are generated and the crossover is performed in the following way:

Chromosome 1	$\left(c_1^{(1)} \quad c_2^{(1)} \quad . \quad . \quad . \quad . \quad c_m^{(1)}\right)$
Chromosome 2	$\left(c_1^{(2)} \quad c_2^{(2)} \quad . \quad . \quad . \quad . \quad c_m^{(2)}\right)$
Offspring 1	$\left(\theta_1 c_1^{(1)} + (1-\theta_1)c_1^{(2)} \quad \theta_2 c_2^{(1)} + (1-\theta_2)c_2^{(2)} \quad . \quad . \quad . \quad . \quad \theta_m c_m^{(1)} + (1-\theta_m)c_m^{(2)}\right)$
Offspring 2	$\left((1-\theta_1)c_1^{(1)} + \theta_1 c_1^{(2)} \quad (1-\theta_2)c_2^{(1)} + \theta_2 c_2^{(2)} \quad . \quad . \quad . \quad . \quad (1-\theta_m)c_m^{(1)} + \theta_m c_m^{(2)}\right)$

Table 4.2. Crossing over by convex linear combination in which random numbers $\theta_i, i = 1,2,...m$ with $0 \le \theta_i \le 1$ are generated and the convex linear combinations are carried out for each corresponding pair of genes to produce two offsprings.

Mutation. Mutation is intended to prevent fast convergence to a local optimum. Mutation operation randomly changes the offspring resulted from crossover. In the case of direct value encoding we replace a number of randomly chosen vector components and replace them with randomly generated numbers within their respective ranges.

Original Chromosome 1	$\left(c_1^{(1)} \quad c_2^{(1)} \quad . \quad . \quad . \quad . \quad c_m^{(1)}\right)$
Original Chromosome 2	$\left(c_1^{(2)} \quad c_2^{(2)} \quad . \quad . \quad . \quad . \quad c_m^{(2)}\right)$
Mutated Chromosome 1	$\left(c_1^{(1)} \quad c_{2,new}^{(1)} \quad . \quad . \quad . \quad . \quad c_m^{(1)}\right)$ **(Note $c_2^{(1)}$ is randomly chosen and replaced by $c_{2,new}^{(1)}$)**
Mutated Chromosome 2	$\left(c_{1,new}^{(2)} \quad c_2^{(2)} \quad . \quad . \quad . \quad . \quad c_{m,new}^{(2)}\right)$ **(Note $c_1^{(1)}$ is randomly chosen and replaced by $c_{1,new}^{(1)}$, so is $c_m^{(1)}$)**

Table 4.3 Illustration of mutation

Selection. There are many methods in selecting the best chromosomes such as the roulette wheel selection, the rank selection, the tournament selection and so on. Elitism, i.e., the copying of the best chromosome (or few best chromosomes) to the new population is usually required to ensure convergence of the algorithm by always keeping the best. In the study, we apply the ranking selection method by sorting the chromosomes by their fitness and selecting the top half of good solutions.

Control parameters

Whether GA is efficient or not highly depends on the algorithm's control parameters. The control parameters available for adjusting the algorithm are the population size, the crossover probability, and the mutation probability. De Jong (1965) made some suggestions based on his observations of the performance of GAs on a bench work of 5 problems where the examples included discontinuities, high dimensionality, noise and multimodality, and suggested that settings of population size 50, crossover rate 60% and mutation rate 0.1% for satisfactory performance over a wide range of problems. However Grefenstette (1986) concluded that a population size 30, a crossover rate 95% and

mutation rate 1% resulted in the best performance when the average fitness of each generation was used as the indicator, while a population size 80, a crossover rate 45% and mutation rate 1% gave the best performance when the fitness of the best individual member in each generation was used as indicator. Some other empiric studies suggest the crossover rate should be high in general and around about 80% ~ 95%, although problem-specific results may suggest other rates (around 60% for example). Mutation rate is problem specific with a range of 1% ~ 10%. The commonly suggested population size is about 20 ~ 30, however sometimes sizes 50 ~ 100 are reported as the best. Some research also shows the best population size depends on the dimension of the search space – the higher the dimension, the larger the population size. In general, it seems these parameters are rather problem specific and should be closely related to the problem at hand. It may be a good practice to try several sets of control parameters and compare the results.

4.1.3 Numerical result using GA

We applied GA to the above described model, using the following specifications:

1. Initial population size 200.
2. Population size of the consecutive generations 40.
3. A mutation rate of 10%.
4. Paring according to the ranks of individuals.
5. Algorithm stopping after it runs for 100 generations.

The following figure shows the path of the best solutions of the 100 generations and the decrease of the cost function in the searching process. We made three runs of the algorithm independently and the paths are shown in a same figure.

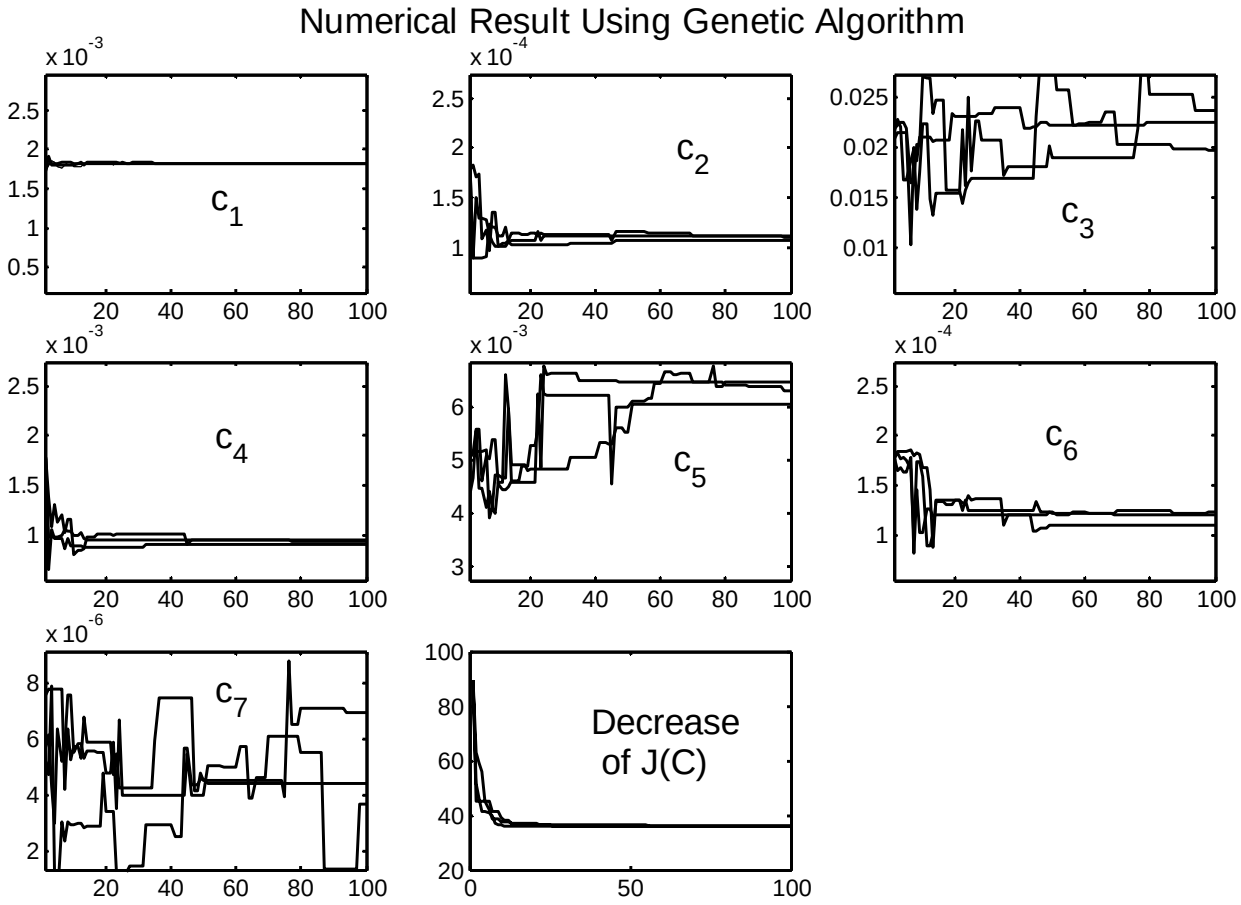

Figure 4.2 Convergence of parameters and decrease of cost function with GA

4.1.4 Discussion of the numerical result

The result is remarkably consistent with the result shown in Figure 3.1. It is seen that even under the situation of noise-disturbed model functions (Figure 2.2), we still obtain the same or even better result (notice the consistent path trends for c_3 and c_5). The cost function is valued at about 35.8 when the algorithm is stopped.

The estimation of the convergent variables is:

$$c^o = \left(1.8224\times10^{-3},\quad 1.1121\times10^{-4},\quad -,\quad 0.9344\times10^{-3},\quad -,\quad 1.1424\times10^{-4},\quad -\right)$$

The estimation of the two parameters for c_3, c_5 are: $0.018 \le c_3 \le 0.025$, $6\times10^{-3} \le c_5 \le 7\times10^{-3}$. The variability of c_7 is the largest and its value cannot be determined.

4.2 Simulated annealing (SA)

4.2.1 General description of SA

The idea of SA was developed by Kirkpatrick et. al. (Kirkpatrick et al. 1983) motivated by the paper published by Metropolis et al. (1953). SA is based on the analogy of global optimization to annealing in solids: a solid is heated to melt and then cooled down; if the cooling is made slowly enough, crystal structures will be perfect (lowest energy state); if the liquid is "quenched" the crystals will contain imperfections. SA simulates this natural cooling mechanism by gradually lowering the temperature of the solid until it converges to a lowest energy state. The following table presents the analogy between the physical annealing process and the SA algorithm (Dowsland 1995):

Physical Annealing	**Optimization**
System States	Feasible Solutions
Energy	Cost
Change of State	Neighbouring Solution generation
Temperature	Control Parameter
Frozen State	Heuristic Solution

Table 4.1 The analogy between the physical annealing process and the SA

SA chooses a random move based on the previous point and allows uphill steps to escape local minima. If the move is better than its current position in terms of minimization then SA will always take it; otherwise it will accept the new move based on some probability.

The following describes the SA algorithm and it is taken from Russell (1995) with a slight modification.

SA algorithm

Function Simulated-Annealing returns a solution state

Inputs : *Problem*: model *F*, admissible set Ω and cost function *J*

Schedule: a sequence of decreasing temperatures

Local Variables: *Current*: a node in Ω

Next: a node in Ω

T: a "temperature" controlling the hill-climbing probability

Current = make-node (initial node)

For t = 1 to ∞ do

T = *Schedule*[t]

If T = 0 then return C*urrent*

Next = a randomly selected successor of *Current*

$\Delta J = J(Next) - J(Current)$

If $\Delta J > 0$ then *Current* = *Next*

else *Current* = *Next* only with probability $\exp(-\Delta J / T)$

The flowchart of SA is shown in Figure 4.3.

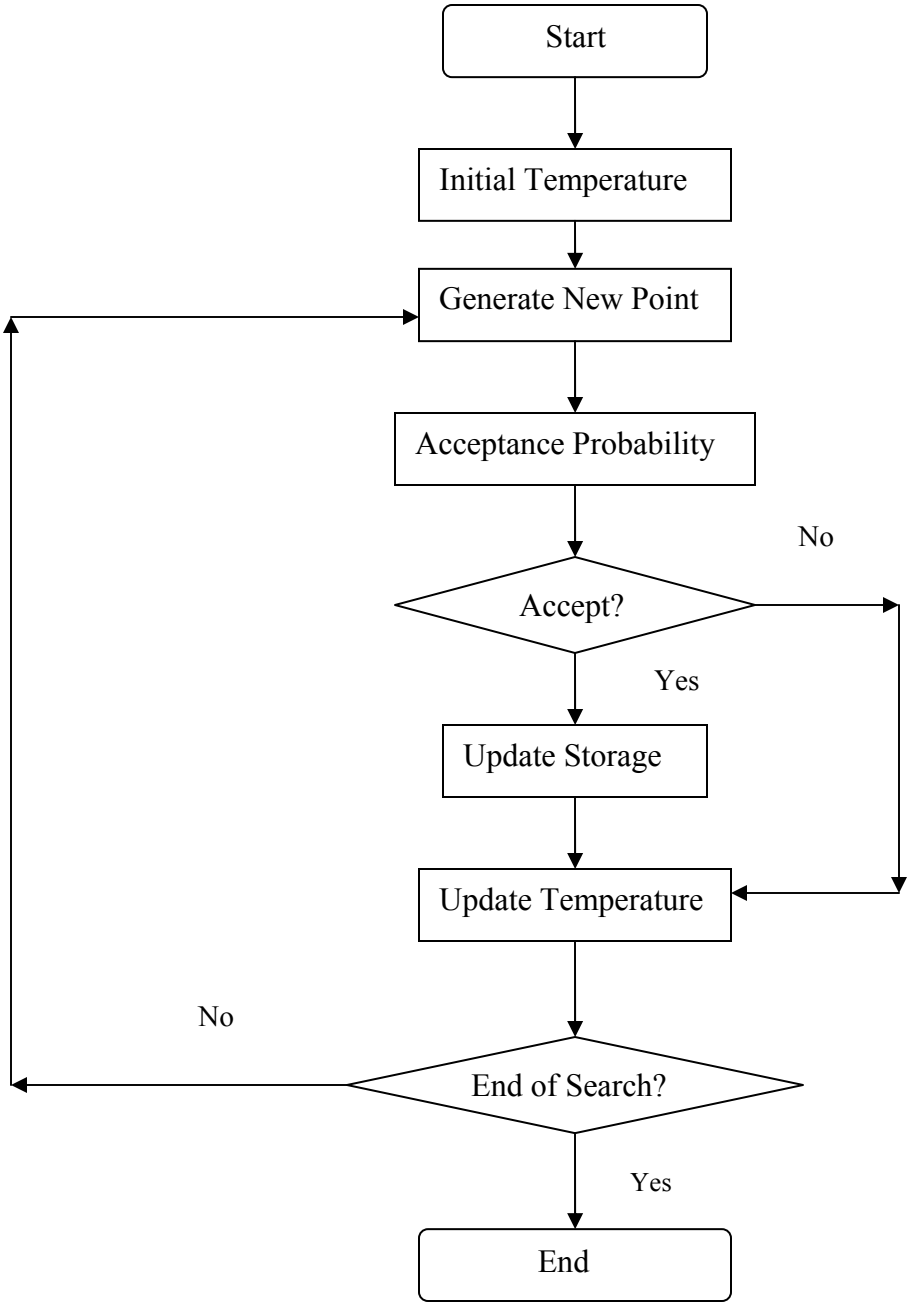

Figure 4.3. Flowchart of SA

The acceptance criterion in SA is based on the law of thermodynamics which states that at temperature t the probability of increasing energy by a magnitude of ΔJ is $p = e^{-\Delta J/kt}$,

where k is Boltzmann's constant (which is usually dropped in SA since it can be contained in parameter t). Thus SA calculates cost function at a new state and accepts the new state whenever the cost is decreased. In the meanwhile, it also accepts new state with probability $p = e^{-\Delta J/kt}$, which gives its hill-climbing ability. It's obvious as t decreases the probability of accepting a hill-climbing move is decreased.

The annealing schedule of SA consists of starting temperature t_0, final temperature t_f, temperature decrement and number of iterations at each temperature. Parameter t_0 should be high enough to allow access to almost any state in the parameter space to prevent the search from terminating locally. However a too high initial temperature tends to turn SA into purely random search. No known method exists for finding a suitable t_0 for a whole range of problems. Some methods suggest (e.g., Rayward & Smith 1996; Dowsland 1995) to start with a high t_0 and cool rapidly until about 60% of uphill solutions are accepted. This gives the real t_0 on which to start cooling down slowly. The final temperature t_f needs not be decreased to zero – stopping criteria can either be a suitably low t_f or a low enough cost function value. Decrement of temperature is critical to the performance of SA. Theoretically, enough iteration should be made at each temperature t so that the system converges to the stationary distribution at that t. In practice, one needs to balance the number of temperatures with the number of iterations. Temperature decrement can be a simple linear method, or a geometric decrement where $t = t\alpha$ where $\alpha < 1$. Based on experience, α usually takes values between $0.8 \sim 0.99$, with better results being found in the higher α at the cost of longer simulation time. A constant number of iterations at each temperature can be used. An alternative is to dynamically change the

number of iterations as the algorithm progresses. At high temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored. At lower temperatures, the number of iterations can be less as the search is already near the global point. One method is to only do one iteration at each temperature, but to decrease the temperature very slowly according to t = t/(1 + βt) where β is a suitably small value (Lundy 1986). In general, we point out that these parameters are rather problem specific. One really needs to adjust the values several times by observing the performance of the algorithm with the problem at hand.

4.2.2 Numerical result

We applied the SA algorithm to the numerical example with the following specified parameter values:

1. Initial annealing temperature: $t_0 = 5$.
2. Annealing schedule: proportionally decreasing with a ratio of 0.95.
3. Number of temperatures used: 200.
4. Number of Monte Carlo simulations at the initial temperature is 300. This number was decreased linearly as temperature decreases.
5. The candidate point was generated in a neighborhood of the currently accepted point according to a uniform distribution. In this example, each component was perturbed uniformly according to:

   $c_j^{candidate} = c_j^{present} + (rand[0,1] - \frac{1}{2}) \times \frac{R_{c_j}}{L}, j = 1,2,...,m$, where $c_j^{present}$ is the j-th component of the current point c in the simulation chain, $rand[0,1]$ was a random

number uniformly distributed over [0,1], R_{c_j} was the physical bound of component c_j, and L was the scale of R_{c_j} that controls the maximum distance allowed. Should a point fall outside the boundary, its symmetric image across the boundary was taken as a replacement.

6. The algorithm was terminated after exhausting all the specified 200 temperatures. (Or one can set the stopping criterion as $e = \mid J(c_{current}) - J(c_{next}) \mid < \delta$).

The parameters above were obtained on a trial and observation basis. The initial temperature t_0 was chosen to be 5 since observation revealed that there was enough freedom for the solution to oscillate in Ω. The temperature decreasing ratio could be between 0.8 and 0.99, and essentially gave the same answer for this problem.

The convergence of the searching process and the decrease of the cost function in the searching process are shown in Figure 4.4.

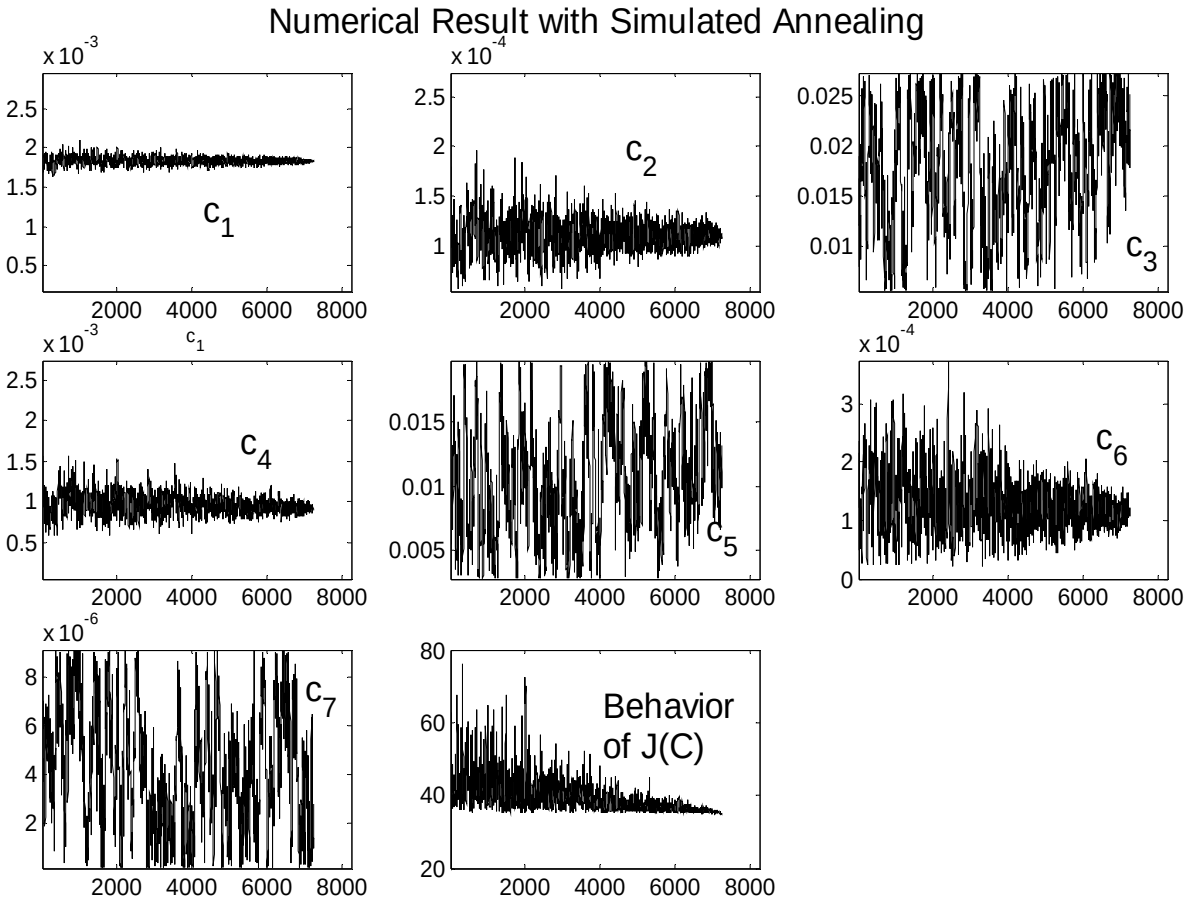

Figure 4.4 Convergence of parameters and decrease of cost function with SA. As can be seen from the behavior of the cost function $J(c)$, high uphill moves are allowed in the initial and middle stages of the search but not at the end of the search when the system is essentially "frozen".

It can be observed that in the searching process up-hill moves were accepted, as there were a large number of increased cost function values. However, as the temperature went

down, the cost function was "frozen" near the lowest value in the sense that large up-hill moves are reduced or essentially eliminated, as is seen from the behavior of $J(c)$.

The result here is also consistent with the above results where conjugate-gradient method and the genetic algorithm were used. The four convergent parameters are c_1, c_2, c_4 and c_6, while c_3, c_5 and c_7 show no sign of convergence. The optimal estimation of SA is

$$c^o = \left(1.8231\times10^{-3},\quad 1.0958\times10^{-4},\quad -,\quad 0.9204\times10^{-3},\quad -,\quad 1.1160\times10^{-4},\quad -\right)$$

and it was calculated by averaging the values of the last 100 accepted samples.

CHAPTER 5

BAYESIAN STATISTICAL INFERENCE AND PARAMETER ESTIMATION

Inverse problems are statistical in nature (Tarantola, 1987). In parameter estimation there always exist estimation errors, or uncertainties due to the statistical nature of the observation as illustrated in Figure (5.1).

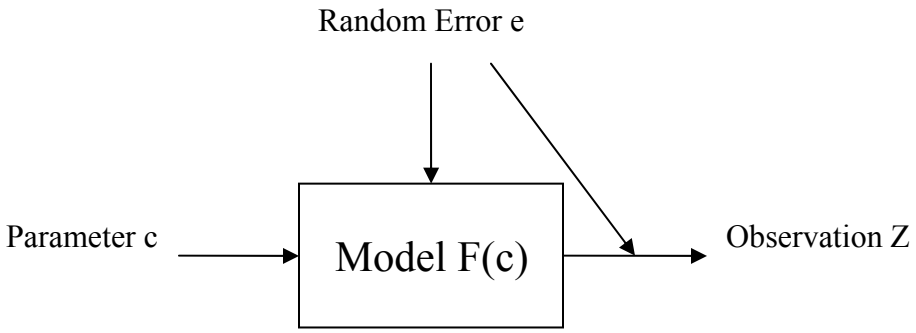

Figure 5.1 Model/Observation with uncertainty

When fitting model parameters, it is always assumed that there is some underlying "true" set of parameters *c* hidden from the experimenter. These true parameters are statistically

realized, along with random measurement errors, as measured data sets symbolized as Z. With the above optimization techniques, Z is treated as a set of fixed values and thus the obtained parameter estimations are also fixed values. In this Chapter, we adopt a probabilistic approach and look at Z as a realization of stochastic processes. It is thus interesting to know the distribution of the parameters if some properties of the randomness in Z are known (or unknown but assumed). Bayesian inversion suits the need by relating forward and inverse probabilities directly (Box and Tiao 1973). In this Chapter, we apply the Bayesian inversion framework to study the inverse problem of Model (2.1) ~ (2.6). Our accomplishments in this Chapter are mainly the following:

1. Giving a brief description of the Bayesian inversion framework and some theoretical background of Markov chain simulation.
2. Proposing the inverse probability of model (2.1) under suitable assumptions.
3. Proposing function approximation approach to study the inverse uncertainty.
4. Using standard MCMC approach to study the inverse uncertainty.
5. Proposing a faster MCMC sampling algorithm specifically for the problem.
6. Calculating the numerical results for the seven-compartmental model and making comparisons about the different approaches.

In general, through a Bayesian inversion approach, we obtain the parameter estimation as has been done in the previous, as well as the quantities that characterize the inverse uncertainty – which is impossible with the previous approaches.

5.1 Forward/inverse probability and Bayes' theorem

We only discuss the case where the uncertainty comes solely from the measurement errors. The uncertainty caused by model structure is also interesting but will be our next stage work. Bayesian statistics provides a theory of inference which enables us to relate the results of observation given in the form of forward probability to parameter uncertainty given in the form of inverse probability. Briefly, the *forward probability* $P(Z \mid c)$ is the conditional probability of the observation data Z given the cause – parameter *c*. The *inverse probability* $P(c \mid Z)$ is the conditional probability of the cause *c* given the observation Z, and it represents the state of knowledge of *c* after measuring *Z*. Bayes' theorem links the two probabilities in the following manner:

$$P(c \mid Z) = \frac{1}{P(Z)} P(Z \mid c) P(c) \tag{5.1}$$

The above equation tells how one should update the state of knowledge of *c* based on the acquired knowledge *Z*. The probability $P(c)$ represents the prior knowledge about *c* before making the observation. The inverse probability $P(c|Z)$ (also called the *posterior probability* of *c*) embodies the posterior knowledge about *c* after making the observation *Z*. The forward probability $P(Z \mid c)$ in (5.1) is usually thought as a function of *c* for fixed observation *Z* and is thus also called the *maximum likelihood function* of *c*. $P(Z)$ is the probability of the occurrence of the data, and is unknown usually. But $P(Z)$ does not affect sampling from $P(c|Z)$ when using the Metropolis-Hastings algorithm, as it is cancelled as a normalizing constant in the algorithm, which we will see later.

5.2 Proposing the inverse probability density function of Model (2.1)

Now we set out to propose the posterior probability density function (PPDF) for the carbon transfer coefficients in Model (2.1). The specification of the PPDF is based on the statistical properties of the observational random errors. Thus for a Bayesian approach, the knowledge of the observational errors is as important as the observational data themselves. However, in practice the information about the probability distributions of errors is often unavailable, with the best that one can have being the errors bars specified by experimenters. In this study, to proceed with our analysis we assume the following:

1. Within each data set $\breve{Z}^k$, $k = 1,2,...,s$, the observation errors E^k, $k = 1,2,...,s$ follow a multivariate Gaussian distribution with a covariance matrix $\mathrm{cov}(E^k)$.
2. The errors among different data sets are independent.

While the second assumption is a physically reasonable one, the first assumption is often used in applications, since due to the Central Limit Theorem, Gaussian distribution is general enough to fully characterize the fluctuation of the combined error effects of various sources (e.g., Von Mises 1964). Another reason for assuming Gaussian-type of error is because maximizing the PPDF formulated under this assumption is equivalent to minimizing the quadratic criterion in the deterministic optimization approach. In the next we denote the prior probability density function as $p_0(c)$ and the PPDF as $p(c)$.

5.2.1 The prior information

In the Bayesian paradigm the prior probability of the model parameter *c* can be regarded as an initial degree of belief in the model parameter *c*. Hence the prior probability $p_0(c)$ represents current state of knowledge about the parameters and can be of any density function form. Lacking of prior knowledge can be represented by a uniform distribution over the admissible set Ω. This is the conceptually simplest non-informative prior (Tarantola 1987). In this case the prior distribution $p_0(c) = const, \forall c \in \Omega$. With a uniform prior, one sees that the maximum a posterior probability (MAP) $p(c)$ becomes the maximum likelihood probability of *c*. Hence using the uniform prior is justified from the point of view of classical (non-Bayesian) statistics.

We can also specify different forms of prior probability distributions for the model parameter *c* based on the current state of knowledge. One way to specify the prior is to use a multivariate Gaussian distribution in which the mean specifies where we believe the parameter is and the variance represents the confidence of the belief. With the Gaussian prior, there is a natural relationship between the confidence of the belief and the Tikhonov regularization number λ that we discussed in Chapter 2.1 (Fitzpatrick 1991).

5.2.2 The posterior probability density function (PPDF) *p*(*c*)

Similarly as in Chapter 2, let $\widetilde{X}(c)$ denote the vector of state variables at all time steps from a finite difference method that solves the equation (2.1), $\Phi^k(c)$ denote the mapping

matrix from $\tilde{X}(c)$ to data set $\breve{Z}^k, k = 1,2,...,s$, and the error covariance matrix of the *k*-th data set be $\text{cov}(E^k)$. Based on the above assumptions (1) and (2), the likelihood function for a single data set $\breve{Z}^k$ is $p(\breve{Z}^k \mid c) \propto \exp\left(-\frac{1}{2}(E^k)^T \text{cov}(E^k)^{-1} E^k\right)$ and the likelihood function of the multiple data sets is $\prod_{k=1}^{s} p(\breve{Z}^k \mid c)$. By specifying the prior probability density function as $p_0(c)$, the PPDF is then

$$p(c) \propto p_0(c) \prod_{k=1}^{s} p(\breve{Z}^k \mid c) = p_0(c) \exp\left(-\sum_{k=1}^{s}\left(-\frac{1}{2}(E^k)^T \text{cov}(E^k)^{-1} E^k\right)\right)$$

i.e.,

$$p(c) \propto p_0(c) \exp\left(-\sum_{k=1}^{s}\frac{1}{2}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right)^T \text{cov}(E^k)^{-1}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right)\right) \tag{5.2}$$

From (5.2) we see if the prior probability $p_0(c)$ of interest is a Gaussian prior $N(c^{prior}, \lambda^{-1}G^{-1})$ (Fitzpatrick 1991), then

$$\begin{aligned} p(c) &\propto \exp\left(-\frac{1}{2}\lambda\left(c - c^{prior}\right)^T G\left(c - c^{prior}\right)\right) \\ &\quad \times \exp\left(-\sum_{k=1}^{s}\frac{1}{2}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right)^T \text{cov}(E^k)^{-1}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right)\right) \\ &= \exp\left(-\sum_{k=1}^{s}\frac{1}{2}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right)^T \text{cov}(E^k)^{-1}\left(\breve{Z}^k - \Phi^k(c)\tilde{X}(c)\right) - \frac{1}{2}\lambda\left(c - c^{prior}\right)^T G\left(c - c^{prior}\right)\right) \end{aligned} \tag{5.3}$$

from which we see the equivalence between the MAP solution of c and the least squares optimal solution of c with Tikhonov regularization specified as in (2.4) of Chapter 2.1. Note that the normalizing constant that makes the above function a probability density function is ignored for the convenience of notation.

Function (5.2) gives a complete probabilistic description of the model parameter c and is thus the inverse problem solution from a Bayesian point of view. However, one needs to analyze it in order to fully display the information hidden inside. In the next, we introduce two approaches – an approximation approach (also called Laplace Approximation in Box & Tiao (1973)) and an MCMC simulation approach – to study the PPDF (5.2) of Model (2.1).

5.3. An approximation approach for inverse uncertainty analysis

Depending on the model structure, the data sets and the admissible parameter space, the shape of the PPDF $p(c)$ is usually hard to determine. However, in cases where $p(c)$ has a single interior dominant mode that falls away from the maximum in an approximately Gaussian fashion, then it is useful to approximate the probability density by Gaussian function (Box & Tiao 1973). The key advantage of this approach is that Gaussian function is completely determined by the mean vector and the covariance matrix. By approximating the PPDF (5.3) with a Gaussian function, we can easily construct marginal distributions for the parameter vector c and calculate the covariance matrix. For this

purpose, we need to use the first and the second derivative information derived in Chapter 2. We adopt the following steps:

1. Find the optimal point c^o for $J(c)$ of (2.4) using the descent-type of algorithm as was done in Chapter 2. This gives the mean of the approximate Gaussian.
2. Expand $J(c)$ as a Taylor series up to the second order around the point c^o. The linear terms vanish since c^o is an extreme. Thus we find

   $$J(c) \cong J(c^o) + \frac{1}{2}(c - c^o)^T Q(c - c^o)$$

   where Q is the Hessian matrix of $J(c)$ derived in Chapter 2, with

   $$Q = (q_{ij}) = \left(\frac{\partial^2 J}{\partial c_i \partial c_j} \right).$$

3. The approximation Gaussian is then

   $$g(c) = \frac{1}{\sqrt{(2\pi)^n \det(Q^{-1})}} \exp\left(-\frac{1}{2}(c - c^o)^T Q(c - c^o) \right)$$

We denote Q^{-1}, the inverse of the matrix Q, as $\text{cov}(c)$ since it represents the covariance matrix of the parameter vector c. Then the PPDF $p(c)$ (5.3) is approximately:

$$p(c) \approx \frac{1}{\sqrt{(2\pi)^n \det(\text{cov}(c))}} \exp\left(-\frac{1}{2}(c - c^o)^T \left(\text{cov}(c)\right)^{-1} (c - c^o) \right)$$

The above Gaussian function has a simple interpretation: the diagonal elements of cov(c) are the variances of the parameters and the off-diagonal elements describes the correlations each pair of parameters. It should be noted that the above approximation is a local approximation and is useful only when the shape of the PPDF can be well-approximated. A perfect approximation only exists in the case of a linear relationship between the data space and the parameter space.

5.4 A stochastic simulation approach for inverse uncertainty analysis – Markov chain Monte Carlo (MCMC)

The approximation approach for inverse uncertainty analysis is not applicable in general since it is a local method. A more general way to study the PPDF (5.3) is to sample it directly and let the information about the parameter c be computed from the samples. The work of Geman and Geman (1984) introduced the Gibbs sampler as a method for obtaining difficult posterior quantities in image restoration. The subsequent integrating article by Gelfand and Smith (1990) facilitated the use of MCMC methods to evaluate integral quantities. The basic idea of MCMC is to design a Markov chain with the probability of interest as the stationary distribution. The Markov chain simulation is run for sufficiently long time (after an initial running time called the *burn-in period*) till samples can be taken from the desired probability distribution. Quantities associated with the distribution are summarized from the large number of samples. A detailed treatment of MCMC can be found in Gamerman (1997).

In the next we first explain the fundamentals of Markov chain simulation, and then we introduce the Metropolis-Hastings (M-H) algorithm. We will also propose a fast sampling algorithm for PPDF (5.2) after the discussion of the M-H algorithm. Only the discrete Markov chain is discussed here, considering the discrete nature of numerical simulation in practice. The contents can be found in most textbooks on Markov process. We refer to lecture notes by Tan, Fox and Nicholls (2004) for some details.

5.4.1 Markov chain

Let $c^{(0)}, c^{(1)}, c^{(2)}, ..., c^{(k)}, c^{(k+1)}, ...$ be a sequence of random variables taking values from a state space $\Omega = \{0, 1, 2, 3, ...,\}$. A Markov chain is a stochastic process with the property that a state $c^{(k+1)}$ in the chain depends only on its immediate previous state $c^{(k)}$ and is thus conditionally independent of all other previous states, i.e., $P(c^{(k+1)} = j \mid c^{(0)} = i_0, ..., c^{(k-1)} = i_{k-1}, c^{(k)} = i) = P(c^{(k+1)} = j \mid c^{(k)} = i)$.

A Markov chain is completely determined by fixing an initial condition $P(c^{(0)} = i_0)$ and the transition probability $P(c^{(k+1)} = j \mid c^{(k)} = i)$. A **homogeneous** Markov chain has the property that $P(c^{(k+n+1)} = j \mid c^{(k+n)} = i) = P(c^{(k+1)} = j \mid c^{(k)} = i)$. The transition probability is usually written as $P_{ij} = P(c^{(k+1)} = j \mid c^{(k)} = i)$ and P_{ij} satisfies the condition $\sum_{j \in \Omega} P_{ij} = 1$. A matrix with rows summing to one is called a **stochastic matrix** (denoted as S).

5.4.2 Stationary distribution

Consider $P(c^{(k)} = j), j = 1,2,...$, the probability that after k steps the probability that the state is at *j*. Let $\pi_j^{(k)} = P(c^{(k)} = j)$ and let $\pi^{(k)} = \left(\pi_1^{(k)} \quad \pi_2^{(k)} \quad . \quad . \quad .\right)$, then it is easily shown $\pi^{(k)} = \pi^{(k-1)} S$ where S denote the stochastic matrix. If there exists π so that $\pi = \pi S$, with $\sum_j \pi_j = 1$, then π is called a **stationary distribution** of the Markov chain. If $\pi^{(k)} \to \pi, k \to \infty$ for any $\pi^{(0)}$ then π is called the **equilibrium distribution** of the chain and the chain is said to be **ergodic**.

For a Markov chain to be ergodic, the following conditions of **ergodicity** must be satisfied:

1. **Irreducibility**. If for any two states *i*, *j* in Ω there is a path of non-zero probability which links *i* to *j* and vice versa, then the chain is said to be irreducible.
2. **Reversibility**. A homogeneous Markov chain is said to be reversible if the transition probability satisfies: $P(c^{(k+1)} = j \mid c^{(k)} = i) = P(c^{(k)} = j \mid c^{(k+1)} = i)$. The necessary and sufficient condition for reversibility is given by the following theorem: let Q be the stochastic matrix of a Markov chain with a unique stationary π. The chain is reversible if and only if the detailed balance condition holds: $\pi_i P_{ij} = \pi_j P_{ji}, i, j \in \Omega$.
3. **Aperiodicity**. Using the set $T = \left\{n : \left(S^n\right)_{ii} > 0, n > 0\right\}$ to define the steps on which it is possible for a chain to revisit *i* once it starts from *I*, if the greatest common devisor of the integers in set *T* is one, the chain is called aperiodic.

We have the following important theorem of ergodicity.

Ergodicity theorem. For an irreducible, aperiodic Markov chain on a countable state space with transition matrix S, if there exists $0 \le \pi_j \le 1, \sum_j \pi_j = 1$ and $\pi_i P_{ij} = \pi_j P_{ji}$ for $i, j \in \Omega$, then the chain is reversible and ergodic with unique stationary distribution π.

The above result answers the existence problem of the equilibrium distribution π of a chain with a given stochastic matrix. The reverse problem is, given a distribution π, how to construct a transitional probability of a Markov chain so that the equilibrium distribution of this chain is π? This has been answered by the MCMC algorithms.

5.4.3 Metropolis-Hastings (M-H) MCMC

M-H MCMC is an algorithm that generates Markov Chain samples from a desired equilibrium distribution. The algorithm is as follows:

Let $c^{(0)}$ be the starting point of the chain and suppose the algorithm has been run to obtain values $c^{(0)}, c^{(1)}, ..., c^{(k-1)}$ in the admissible set Ω. The next point of the chain $c^{(k)}$ is obtained by a two-step process consisting of a "proposal step" and a "move step".

Proposal step: Propose a candidate state c in the admissible set on the basis of $c^{(k-1)}$ according to some proposal probability $q(c \mid c^{(k-1)})$ where $q(c \mid c^{(k-1)})$ satisfies:

1. $q(c \mid c^{(k-1)}) = 0 \Leftrightarrow q(c^{(k-1)} \mid c) = 0$.

2. $q(c \mid c^{(k-1)})$ forms the transition matrix of an irreducible Markov chain on Ω.

Move step: With probability

$$p(c \mid c^{(k-1)}) = \min\left\{1, \frac{\pi(c)q(c^{(k-1)} \mid c)}{\pi(c^{(k-1)})q(c \mid c^{(k-1)})}\right\}$$

accept

$c^{(k)} = c$ (acceptance)

else set

$c^{(k)} = c^{(k-1)}$ (rejection)

Thus the transition probability is given by $p(c \mid c^{(k-1)}) = q(c \mid c^{(k-1)})\, p(c \mid c^{(k-1)})$. It is shown that the Markov chain simulated by the M-H algorithm is reversible with respect to π. If the Markov chain is also irreducible and aperiodic, then it is an ergodic Markov chain with unique equilibrium distribution π (Metropolis et al. 1953; Hastings 1970).

Note the algorithm is presented in two steps. In the first step, a candidate state *c* is generated with the proposal distribution $q(. \mid c^{(k-1)})$. The second step ensures that the next sample lies in the high probability region of π. Thus to generate the next Markov chain sample on the basis of the current sample, one first generates a candidate state, and then takes either the candidate state or the current sample as the next sample according to whether the candidate state lies in the high probability region or not.

5.4.4 MCMC estimator and output analysis

From the Bayesian point of view, the complete state of knowledge of the parameter vector *c* after the observation *Z* is given by the PPDF $p(c)$. Once $p(c)$ is known, various quantitative estimates can be made such as the maximum likelihood estimates, the MAP estimates, the mean estimates and the variances/covariances of the variables etc. can be calculated. For example:

1. the mean estimator $E(c) = \int cp(c)dc$
2. the parameter covariance matrix $Cov(c) = \int (c - E(c))(c - E(c))^T p(c)dc$
3. the marginal distribution $p(c_i) = \int p(c)dc_1 ... dc_{i-1} dc_{i+1} ... dc_m$

In general one needs to calculate the following form of integral:

$$E[f(c)] = \int f(c)\, p(c)dc$$

We can estimate this integral from some independent and identically distributed (i.i.d.) samples of the random variable *c*, $c^{(1)}, c^{(2)}, ..., c^{(k)}$, from $p(c)$, i.e.,

$$E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}] = \frac{1}{k} \sum_{i=1}^{k} f(c^{(i)}) .$$

If the samples $c^{(i)}, i = 1,2,...,k$ are i.i.d., the strong and weak laws of large numbers ensure that the approximation can be made accurate as desired with increasing k. It should be noted $E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}]$ is a random variable itself. For different realizations $c^{(1)}, c^{(2)}, ..., c^{(k)}$ of the random variable *c*, the estimation will be different. But in general,

if $c^{(1)}, c^{(2)}, ..., c^{(k)}$ were independent samples and $E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}] = \frac{1}{k}\sum_{i=1}^{k} f(c^{(i)})$, then $\mathrm{var}\left(E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}]\right) = \frac{\mathrm{var}(f(c))}{k}$, which gives the convergence rate of the estimator.

In MCMC, the samples are dependent on their immediate predecessors and hence $c^{(1)}, c^{(2)}, ..., c^{(k)}$ are correlated samples. In this case and we have the following estimate of the variance for $E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}]$:

$$\mathrm{var}\left(E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}]\right) = \frac{\tau_f \, \mathrm{var}(f(c))}{k}$$

where τ_f is called the autocovariance time and is given by:

$$\tau_f = 1 + 2\sum_{s=1}^{\infty} \rho_{ff}(s)$$

and $\rho_{ff}(s)$ is the normalized autocovariance function

$$\rho_{ff}(s) = \mathrm{cov}\left(f(c^{(k)}), f(c^{(k+s)})\right) / \mathrm{var}(f).$$

Thus the MCMC estimators should be phrased in the following way: we estimate the mean value of $f(c)$ in the interval of $E[f(c) \mid c^{(1)}, c^{(2)}, ..., c^{(k)}] \pm 2 \times \sqrt{\frac{\tau_f \, \mathrm{var}(f(c))}{k}}$ with 95% confidence. It can also be seen that if a Markov chain is well designed such that the autocovariance time is small, then the estimators will converge fast.

5.5 Toward a more efficient sampling method

The above M-H algorithm is quite general. Usually the computation is quite intensive if the proposal probability is not properly set, as a large number of samples will fall out of the high probability density area and are thus rejected. When directly applied to Model (2.1), the computation is extremely time-consuming since each simulation involves the forward solution of the system equation over a long time span, especially when the finite difference method is on a fine grid. A successful application of MCMC relies on a proper choice of the proposal probability, as the proposal probability not only affects the distribution of the candidate state given the present state, but also the convergence rate of the MCMC estimators. In the following we discuss two major types of proposal distribution and reveal the mechanism of how to improve the efficiency of the MCMC algorithm.

5.5.1 Symmetric proposal probability

One common type of proposal probability is the symmetric distribution centered at the current state. In this case $q(c \mid c^{(k-1)}) = q(c^{(k-1)} \mid c)$. This proposal probability is also called "chain-adaptive" since the distribution of the candidate state depends on the current state. The most commonly used symmetric distribution can either be symmetric Gaussian or uniform distribution. However, the step lengths $l_i, i = 1,2,....,n$ for a proposal probability in n-dimensional space that control the allowable distance that the next sample can depart from the current one affects the sampling efficiency. In general, small step lengths tend to increase the correlation among the Markov chain samples, slowing down the convergence of the MCMC simulation, while large step values will increase the number of rejected

samples and thus also slow down the convergence of the MCMC simulation. When the step lengths are excessively large in the parameter space, the candidate states will often be generated far away from the current sample, and so may not have a high probability of lying in the high probability region even if the currently-accepted one does. The choice of the step lengths corresponding to the "spreading" of the proposal probability is a trade off between the degree of neighborhood correlations and number of rejected samples.

5.5.2 Stationary proposal probability

As another extreme, the proposal probability can be "stationary" in the sense $q(c \mid c^{(k-1)}) = \chi(c)$ so that the next sample is generated from some pre-specified probability $\chi(c)$ directly sample and thus does not depend on the current one. The constructed proposal probability $\chi(c)$ is usually based on some prior knowledge about the PPDf and is chosen as close to the PPDF as possible. For example, suppose $\chi(c) = p(c)$. Then the acceptance probability becomes:

$$\min\left\{1, \frac{p(c)\,p(c^{(k-1)} \mid c)}{p(c^{(k-1)})\,p(c \mid c^{(k-1)})}\right\} = \min\left\{1, \frac{p(c)\,p(c^{(k-1)})}{p(c^{(k-1)})\,p(c)}\right\} = 1$$

Thus all generated samples will be accepted. In this case, all samples will be i.i.d. according to $p(c)$. This observation shows in order to reduce the computational cost, we should sample the PPDF by utilizing information regarding the PPDF, i.e., we should let the proposal probability be as close to the PPDF as possible.

5.5.3 The proposal probability that increases the sampling efficiency

As was analyzed in Chapter 5.3, the Hessian matrix contains the information about the geometry of the PPDF. This information can be utilized to improve the efficiency of sampling. Thus we propose the following fast Metropolis-Hastings (M-H) type of algorithm:

Denote the inverse of the Hessian matrix at the optimal point as $\mathrm{cov}(c^o)$ similarly as above. Determine eigenvalues γ_i and eigenvectors q_i of $\mathrm{cov}(c^o)$, combine the *m* column vectors q_i into a matrix *T* that diagonalizes the matrix $\mathrm{cov}(c^o)$ as

$$T^{-1}\,\mathrm{cov}(c^o)T = \begin{pmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_m^2 \end{pmatrix}$$

1. *Generate independent Gaussian variables:* Generate *m* mutually independent Gaussian random variables $y_i, i = 1,..., m$ with the variances $\sigma_i^{\,2} = \gamma_i, i = 1,..., m$.
2. *Transform step:* Transform the vector $y = (y_1 \;\; . \;\; . \;\; y_m)^T$ into the correlated Gaussian vector $z = (z_1 \;\; . \;\; . \;\; z_m)^T$ according to $z = T \cdot y$. Then *z* is a sample of a Gaussian random vector with zero mean and covariance $\mathrm{cov}(c^o)$.
3. *Proposal step:* propose the next point according to $c = c^{(k-1)} + z$. Then *c* is a sample of a Gaussian random vector with mean $c^{(k-1)}$ and covariance $\mathrm{cov}(c^o)$.
4. *Move step*: With probability

$$p(c \mid c^{(k-1)}) = \min\left\{1, \frac{p(c)q(c^{(k-1)} \mid c)}{p(c^{(k-1)})q(c \mid c^{(k-1)})}\right\}$$

accept

$$c^{(k)} = c \qquad \text{(acceptance)}$$

else set

$$c^{(k)} = c^{(k-1)} \qquad \text{(rejection)}$$

It is obvious that the above method is chain-adaptive, but it also utilizes the information about the shape of the approximation Gaussian constructed above. The two requirements (1) and (2) on proposal probability in the M-H algorithm are not violated by the added steps. This proposed algorithm increases tremendously the efficiency of sampling, as is evidenced by our numerical results.

5.6 Application to the seven-pool model and numerical results

In the following we give three sets of results for analyzing the PPDF. The first inversion result is from direct function approximation approach. The second comes from the general M-H algorithm where the proposal probability is a uniform distribution. The third is from the modified M-H algorithm where the Hessian matrix information is used in the proposal probability. Throughout the analysis, the prior distribution of the model parameters are assumed to be uniform on the admissible set Ω given in (2.7).

5.6.1 Numerical result of the approximation approach

From the previous discussion, it is seen that we can possibly obtain salient information for parameters c_1, c_2, c_4 and c_6 only. Thus in this section only their joint probability density function is approximated. The procedures are given in the following:

1. From the conjugate gradient method, we have the optimal point: $c^o = \left(1.8238\times10^{-3},\ \ 1.0933\times10^{-4},\ \ -,\ \ 0.9862\times10^{-3},\ \ -,\ \ 1.1471\times10^{-4},\ \ -\right)$.
2. We calculate the Hessian matrix at c^o. The elements corresponding to c_1, c_2, c_4 and c_6 are taken to form matrix Q mentioned in Chapter 5.3.

$$Q = \begin{pmatrix} 9.6316\times10^{8} & 2.8857\times10^{8} & -1.8025\times10^{7} & -3.5738\times10^{7} \\ 2.8857\times10^{8} & 2.0979\times10^{10} & -1.8625\times10^{9} & -7.7445\times10^{8} \\ -1.8025\times10^{7} & -1.8625\times10^{9} & 3.5445\times10^{8} & -1.7179\times10^{8} \\ -3.5738\times10^{7} & -7.7445\times10^{8} & -1.7179\times10^{8} & 1.5968\times10^{9} \end{pmatrix}.$$

3. The inverse of Q is the covariance matrix of c_1, c_2, c_4 and c_6. It is given by

$$\mathrm{cov}(c_1, c_2, c_4, c_6) = \begin{pmatrix} 1.043\times10^{-9} & -1.615\times10^{-11} & -2.560\times10^{-11} & 1.274\times10^{-11} \\ -1.615\times10^{-11} & 1.074\times10^{-10} & 6.209\times10^{-10} & 1.185\times10^{-10} \\ -2.560\times10^{-11} & 6.209\times10^{-10} & 5.556\times10^{-9} & 1.007\times10^{-9} \\ 1.274\times10^{-11} & 1.185\times10^{-10} & 1.007\times10^{-9} & 7.923\times10^{-10} \end{pmatrix}$$

4. As was discussed in Chapter 5.3, the approximation of the PPDF of c_1, c_2, c_4 and c_6 is then given by

$$p(c) \approx \frac{1}{\sqrt{(2\pi)^n \det(\mathrm{cov}(c_1, c_2, c_4, c_6))}} \times$$

$$\exp\left(-\frac{1}{2} \begin{pmatrix} c_1 - c_1^0 & c_2 - c_2^0 & c_4 - c_4^0 & c_6 - c_6^0 \end{pmatrix}^T \left(\mathrm{cov}(c_1, c_2, c_4, c_6)\right)^{-1} \begin{pmatrix} c_1 - c_1^0 \\ c_2 - c_2^0 \\ c_4 - c_4^0 \\ c_6 - c_6^0 \end{pmatrix} \right)$$

5. From the above multivariate Gaussian distribution, it is easy to write the marginal distribution for each of the parameters and also the joint distributions for two or more parameters. The marginal distributions for the parameters are given as:

$$f(c_i) = \frac{1}{\sqrt{2\pi\,\mathrm{var}(c_i))}} \exp\left(-\frac{1}{2}(c - c_i^o)\left(\mathrm{var}(c_i)\right)^{-1}(c - c_i^o) \right), i = 1,2,3,4.$$

The joint distributions for any two parameters are:

$$f(c_i, c_j) = \frac{1}{\sqrt{(2\pi)^2 \det(\mathrm{var}(c_i, c_j))}}$$
$$\times \exp\left(-\frac{1}{2} \begin{pmatrix} c - c_i^o & c - c_j^o \end{pmatrix} \left(\mathrm{var}(c_i, c_j)\right)^{-1} \begin{pmatrix} c - c_i^o & c - c_j^o \end{pmatrix}^T \right), \; i, j = 1,2,3,4.$$

6. The correlation coefficient matrix (i.e., the normalized covariance matrix) reveals the degree of correlations among the coefficients. Based on $\mathrm{cov}(c_1, c_2, c_4, c_6)$, it is approximately:

$$\left(\frac{\mathrm{var}(c_i, c_j)}{\sqrt{\mathrm{var}(c_i) \times \mathrm{var}(c_j)}} \right)_{i,j=1,\ldots,4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.739 & 0.406 \\ 0 & 0.739 & 1 & 0.440 \\ 0 & 0.406 & 0.440 & 1 \end{pmatrix}$$

From the correlation matrix, we see c_1 has no correlation with other parameters. The correlation between c_2 and c_4 is strong and is about 0.739. The correlation between c_2 and c_6 is about 0.40, and the correlation between c_4 and c_6 is about 0.44.

The information of the PPDF is revealed more intuitively by the following figures of marginal distributions and joint distributions:

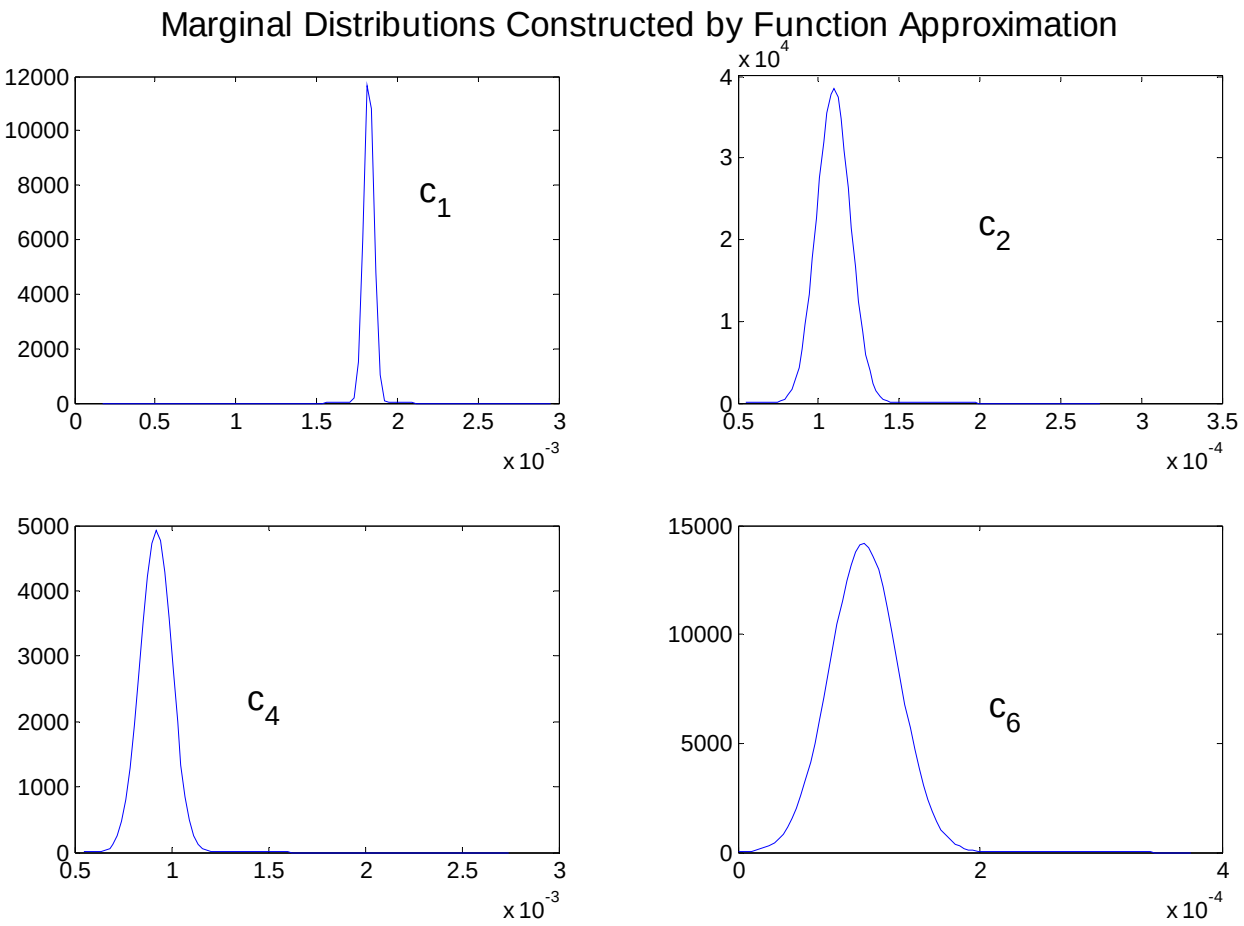

Figure 5.2 Marginal distribution via function approximation approach. The x-axes represent the pre-specified physical bounds for the parameters. From the figures we c_1 has the least uncertainty on its physical range. Parameter c_6 has the largest uncertainty.

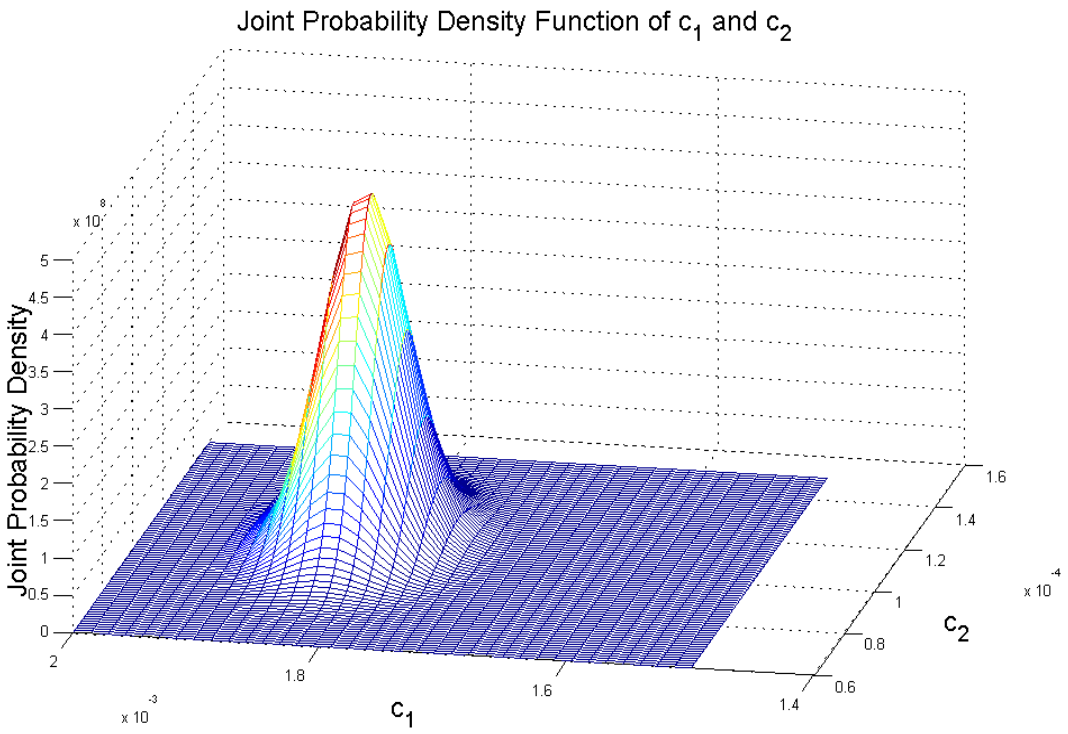

Figure 5.3 Joint distribution of c_1 and c_2 via function approximation approach. As can be seen, since these two parameters are independent, the principal directions of the ellipsoid are parallel to the axes.

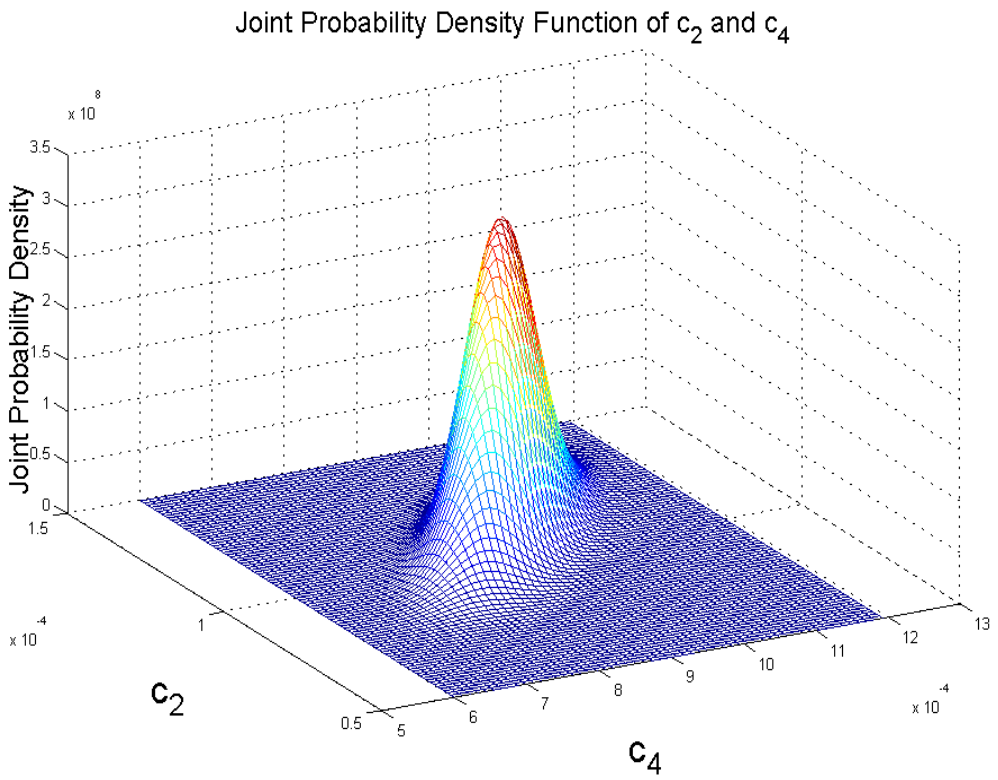

Figure 5.4 Joint distribution of c_2 and c_4 via function approximation approach. The skewness of the principal direction is caused by the correlation.

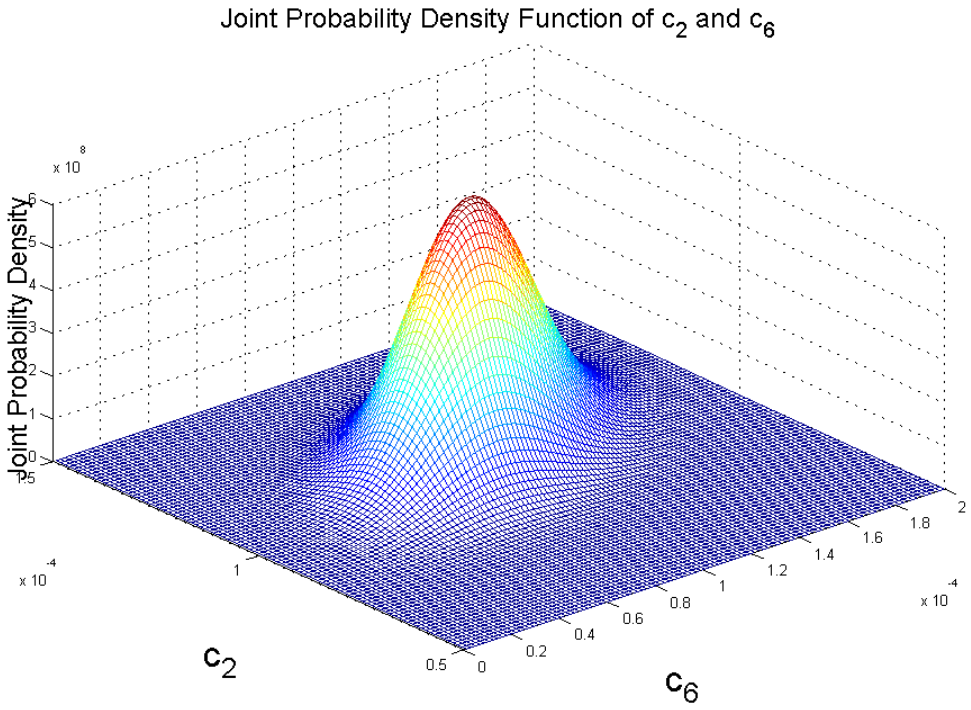

Figure 5.5 Joint distribution of c_2 and c_6 via function approximation approach

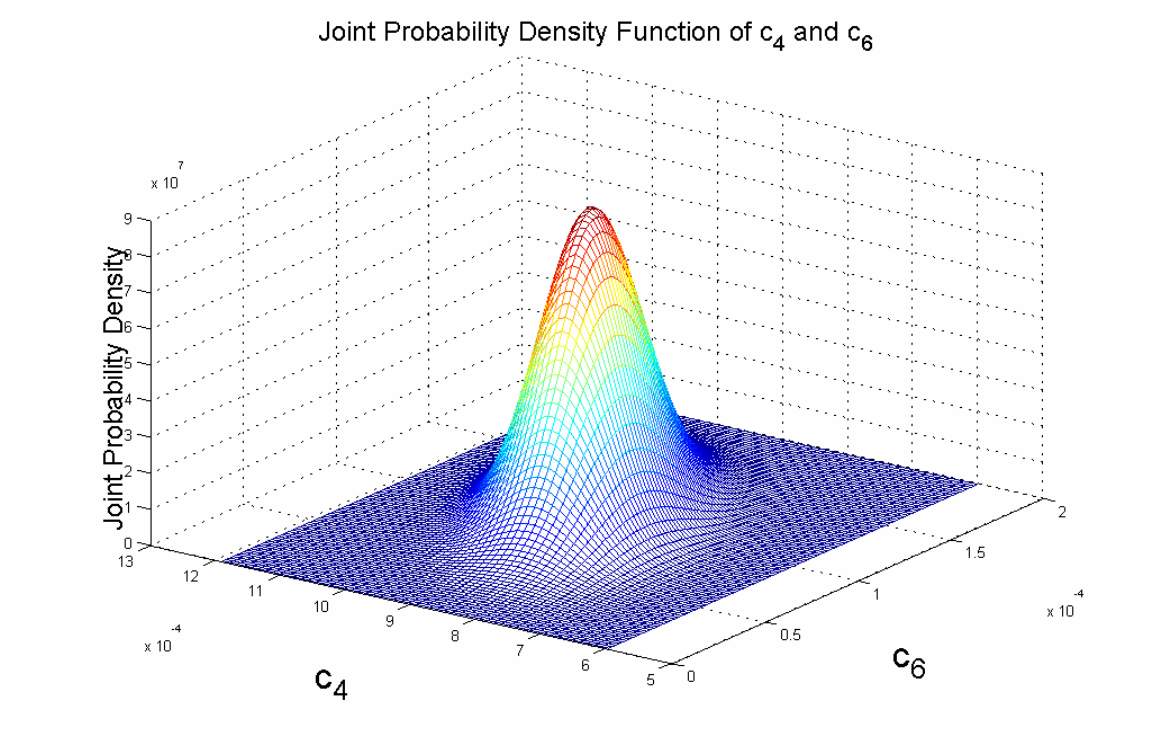

Figure 5.6 Joint distribution of c_4 and c_6 via function approximation approach

From Figure 5.2, it can be seen that parameter c_1 has the least inverse uncertainty, as its marginal distribution is constrained in a very narrow range within the pre-specified

physical bound. The constraints of c_2 and c_4 are about the same. Parameter c_6 has the largest uncertainty as its marginal distribution has the widest spread over its physical bound. The independence of c_1 and c_2 is characterized by the fact that the principal directions of the joint probability function are parallel to the axes. The degrees of correlations between each pairs of parameters are characterized by the skewness of the principal directions relative to the axes.

The following table summarizes the uncertainty in parameter estimation:

Parameters	c_1	c_2	c_3	c_4
Mean	1.8238×10^{-3}	1.0933×10^{-4}	0.9862×10^{-3}	1.1471×10^{-4}
Standard deviation	3.2296×10^{-5}	1.0363×10^{-5}	7.4539×10^{-5}	2.8148×10^{-5}

Table 5. 1 Inverse uncertainty analysis of parameters c_1, c_2, c_4 and c_6

As a conclusion, we see the inverse uncertainty can be obtained from an approximation approach to the PPDF. However, it should be noted that since these constructions are based on the second order approximation of the cost function locally at the optimal point and the Hessian only gives information about the curvature near the optimal point, the characterization of the inverse uncertainty may not be complete if the approximation is not good enough. For example, the variances of the parameters may be underestimated and the correlations may be overestimated, depending on the actual shape of PPDF. However, this approach still reveals most of the information in the PPDF, and, in cases

when cost functions are nearly quadratic with respect to the model parameters, this approach is quite accurate.

5.6.2 Numerical result of the MCMC approach with the standard M-H algorithm

In this section we use M-H algorithm to directly sample the PPDF. By analyzing the samples, we will quantify the inverse uncertainty described by the PPDF. Since the samples follow the stationary distribution given by the PPDF $p(c)$, the direct analyzing of the samples tends to overcome the weakness of the above approximation approach and give a more complete view of $p(c)$ from the samples.

5.6.2.1 Specifications of the MCMC sampling using the standard M-H algorithm

The following are the specifications for the simulation:

1. Five runs of the M-H algorithm were made, each with a simulation time of 20,000.
2. Each run was started at a randomly chosen point in the admissible set.
3. The candidate point was generated according to a uniform distribution. Each component was perturbed uniformly according to:

   $$c_j^{candidate} = c_j^{present} + (rand[0,1] - \frac{1}{2}) \times \frac{R_{c_j}}{L}, \; j = 1,2,...,m,$$

   where $c_j^{present}$ is the j-th component of the current point c in the simulation chain, $rand[0,1]$ is a random number uniformly distributed over [0,1], R_{c_j} is the

physical bound of component c_j, and L is the scale of R_{c_j} that controls the maximum distance allowed. If $c_j^{candidate}$ falls outside the admissible set, then it is reflected back across the boundary.

4. The burn-in length was set to be the first 100 samples.
5. Each run rejected about 18800 candidate samples. The rejection rate was about 94% on the average. Thus only 6% new updates were accepted.
6. The samples of the five runs were mixed to get a total of 99500 samples.

The following figure shows the mixed chains for all the seven parameters.

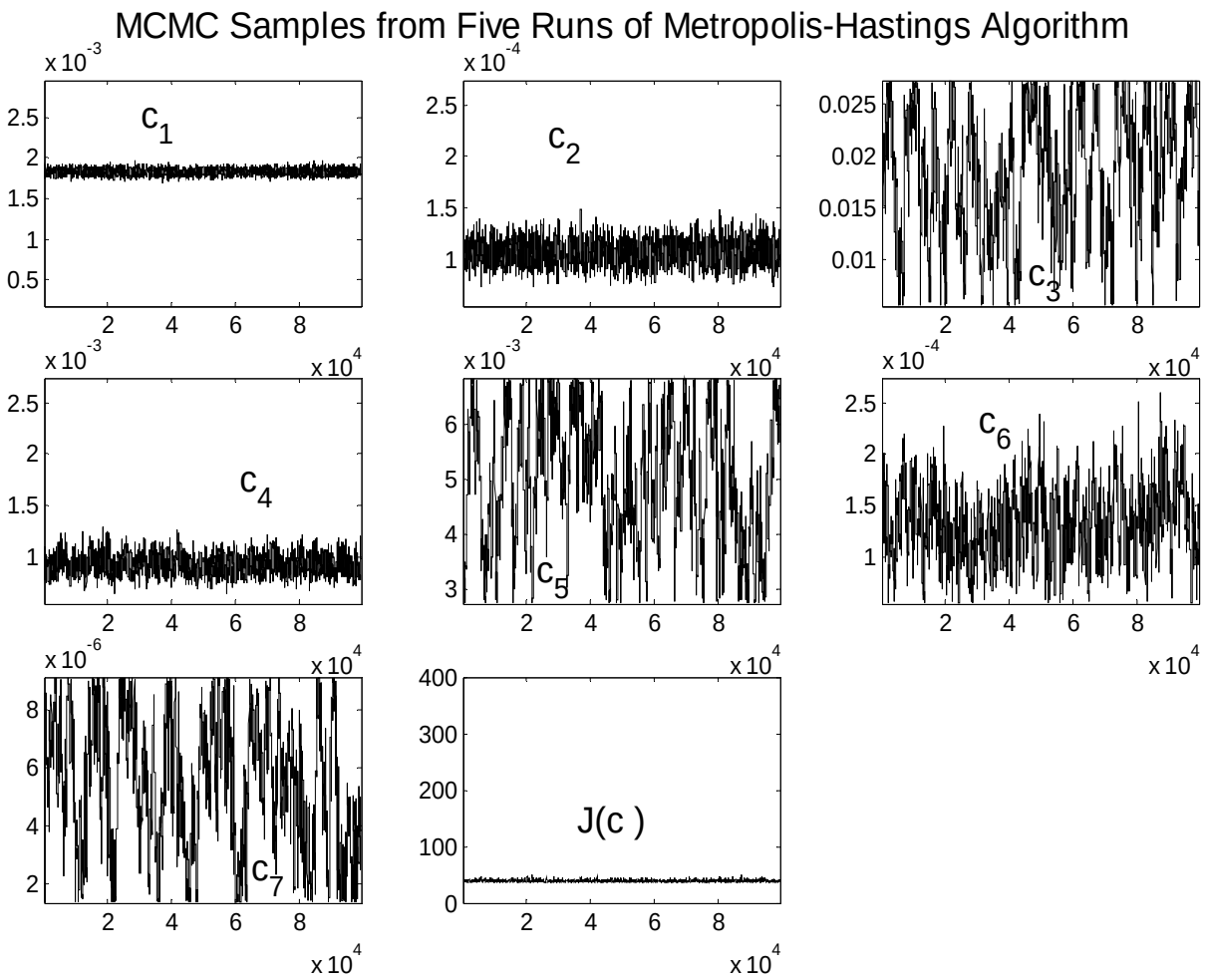

Figure 5.7 MCMC sampling processes for the model parameters. The last plot shows the values of J(c) at the high probability density area.

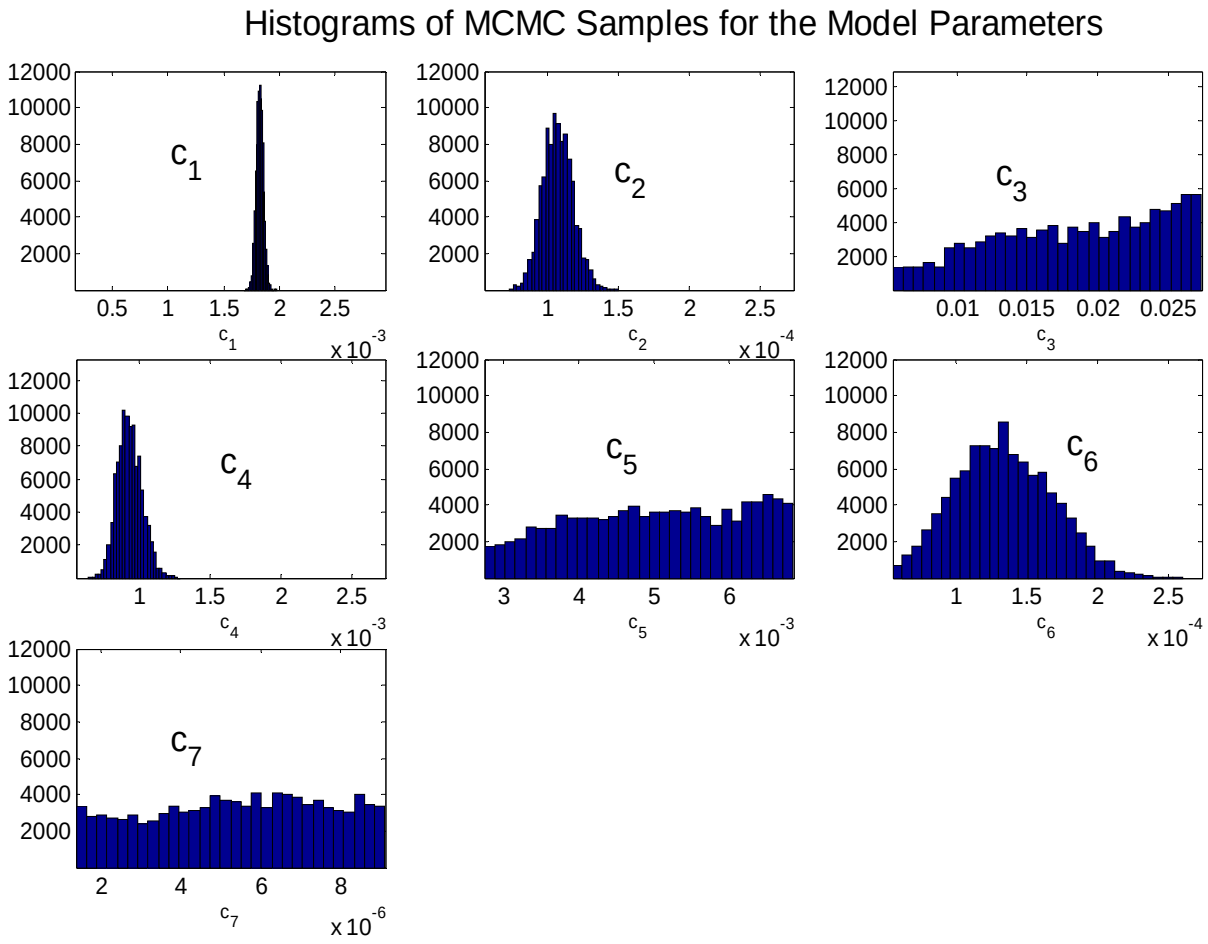

Figure 5.8 Histograms of MCMC samples for the model parameters.

5.6.2.2 Output analysis

1. Mean estimators.

   Based on the mixed sampling series, the means of c_1, c_2, c_4, c_6 are

   $$E[c_1] = 1.8242 \times 10^{-3}, E[c_2] = 1.0669 \times 10^{-4}, E[c_4] = 0.9360 \times 10^{-3}, E[c_6] = 1.2729 \times 10^{-4}$$

2. Covariance matrix.

   The following covariance matrix and correlation coefficient matrix are calculated directly from the samples:

$$\mathrm{cov}(c_1, c_2, c_4, c_6) = \begin{pmatrix} 1.0644 \times 10^{-9} & -8.6965 \times 10^{-12} & -2.1682 \times 10^{-11} & 2.2314 \times 10^{-11} \\ -8.6965 \times 10^{-12} & 1.0835 \times 10^{-10} & 5.4586 \times 10^{-10} & 1.1949 \times 10^{-10} \\ -2.1682 \times 10^{-11} & 5.4586 \times 10^{-10} & 7.5259 \times 10^{-9} & 9.2642 \times 10^{-10} \\ 2.2314 \times 10^{-11} & 1.1949 \times 10^{-10} & 9.2642 \times 10^{-10} & 1.0323 \times 10^{-9} \end{pmatrix}$$

3. Matrix of correlation coefficients.

$$\left(\frac{\mathrm{var}(c_i, c_j)}{\sqrt{\mathrm{var}(c_i) \times \mathrm{var}(c_j)}} \right)_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.6045 & 0.3573 \\ 0 & 0.6045 & 1 & 0.3324 \\ 0 & 0.3573 & 0.3324 & 1 \end{pmatrix}$$

5.6.3 MCMC with fast sampling method

The sampling efficiency of the standard M-H algorithm used above is very low, as is seen from the high rejection rate (about 94%) of the proposed candidate samples. This means 94% of the forward simulations were wasted. This imposes a costly computational burden for computer simulation. In this section, we test the fast M-H-G type of algorithm

and make comparisons with the above approach. The followings are the specifications and numerical results.

5.6.3.1 Specifications of the MCMC sampling

The following are the specifications for the simulation:

1. We run the M-H algorithm with 20,000 simulation times.
2. The candidate point was generated according to the following: c_1, c_2, c_4 and c_6 were generated by the method discussed in Chapter 5.5.2, while c_3, c_5 and c_7 were uniformly generated according to: $c_j^{candidate} = c_j^{present} + (rand[0,1] - \frac{1}{2}) \times \frac{R_{c_j}}{L}, j = 3,5,7.$ similarly as above.
3. The burn-in length was set to be the first 100 samples.
4. The algorithm rejected about 12,000 candidate samples, giving a rejection rate of 60%. Compared with the above rate of 94%, it is a tremendous decrease. Now 40% new updates were accepted due to the fast sampling method.

The following figure shows the sampling chains for all the seven parameters:

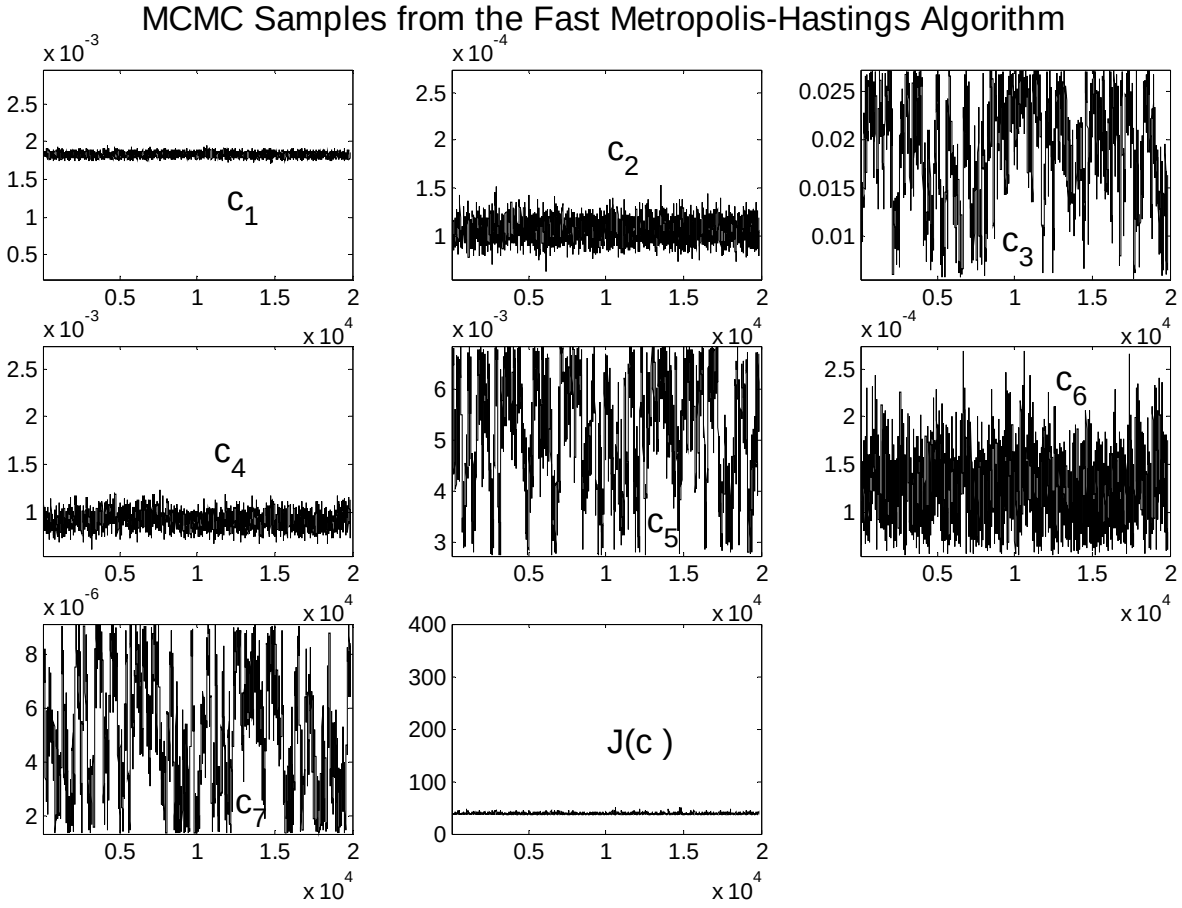

Figure 5.9 MCMC sampling processes with the fast Metropolis-Hastings algorithm

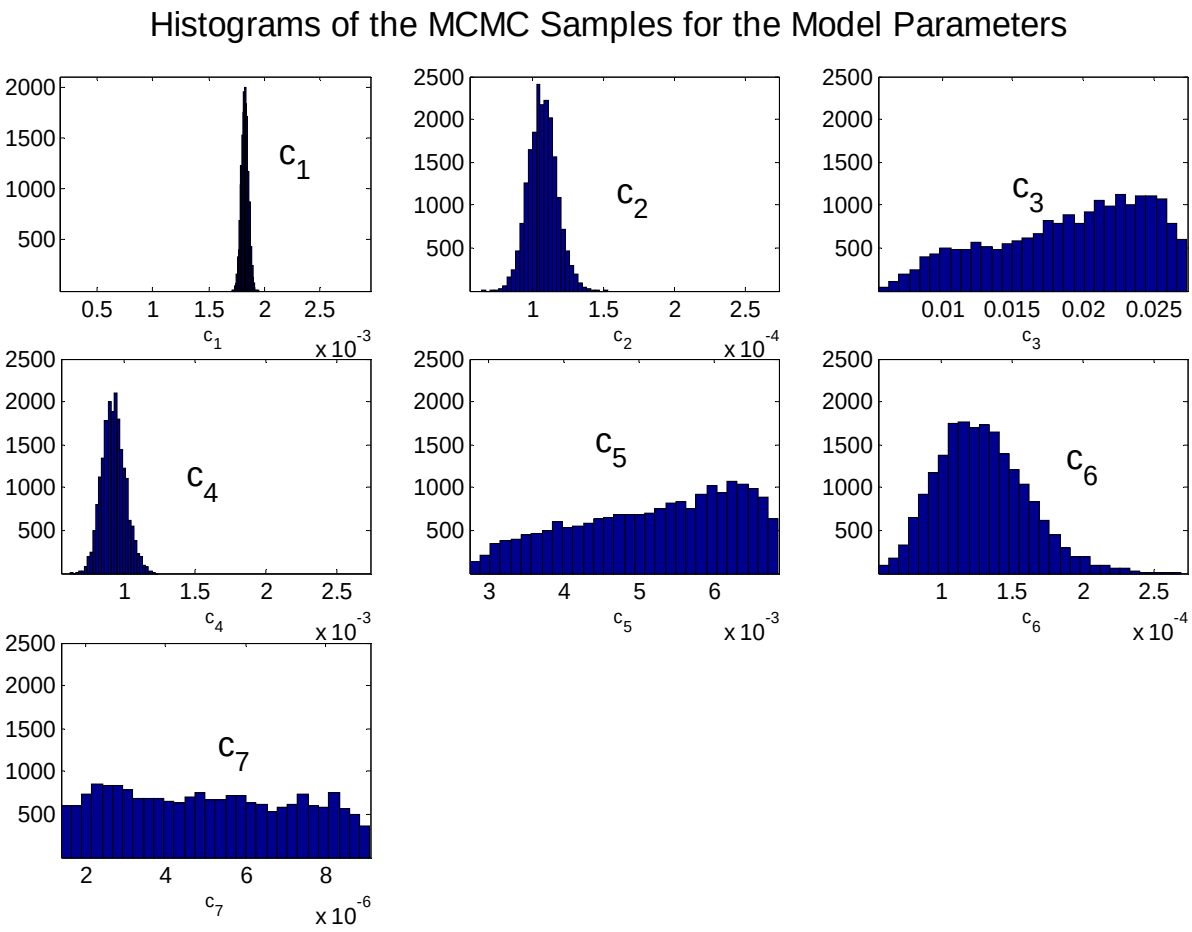

Figure 5.10 Histograms of the samples from fast MCMC

5.6.3.2 Output analysis

Based on the above samples, we calculate the following:

1. Mean estimator

$$E[c_1] = 1.8258 \times 10^{-3}, E[c_2] = 1.0660 \times 10^{-4}, E[c_4] = 0.9337 \times 10^{-3}, E[c_6] = 1.2881 \times 10^{-4}$$

2. Covariance matrix

$$\mathrm{cov}(c_1, c_2, c_4, c_6) = \begin{pmatrix} 1.0650 \times 10^{-9} & -1.9761 \times 10^{-11} & -6.0669 \times 10^{-11} & 3.9704 \times 10^{-11} \\ -1.9761 \times 10^{-11} & 1.0832 \times 10^{-10} & 5.763 \times 10^{-10} & 1.0594 \times 10^{-10} \\ -6.0669 \times 10^{-11} & 5.763 \times 10^{-10} & 6.8488 \times 10^{-9} & 8.174 \times 10^{-10} \\ 3.9704 \times 10^{-11} & 1.0594 \times 10^{-10} & 8.174 \times 10^{-10} & 1.0256 \times 10^{-9} \end{pmatrix}$$

3. Matrix of correlation coefficients

$$\left(\frac{\mathrm{var}(c_i, c_j)}{\sqrt{\mathrm{var}(c_i) \times \mathrm{var}(c_j)}} \right)_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0.6691 & 0.3178 \\ 0 & 0.6691 & 1 & 0.3084 \\ 0 & 0.3178 & 0.3084 & 1 \end{pmatrix}$$

5.7. Result comparison

The above three sets of results are summarized in the following Table 2. From the table, it can be observed that the approximation approach overestimated the correlations among the parameters and underestimated the variance of c_6. However, the estimates are in general quite consistent.

	Approaches		
	Approximation	Standard M-H	Fast M-H
$Var(c_1)$	1.043×10^{-9}	1.0644×10^{-9}	1.0650×10^{-9}
$Var(c_2)$	1.074×10^{-10}	1.0835×10^{-10}	1.0832×10^{-10}
$Var(c_3)$	5.556×10^{-9}	7.5259×10^{-9}	6.8488×10^{-9}
$Var(c_4)$	7.923×10^{-10}	1.0323×10^{-9}	1.0256×10^{-9}

Table 5.2 Comparison of Variance Estimations

	Approaches		
	Approximation	Standard M-H	Fast M-H
c_1, c_j (j = 2,4,6)	0	0	0
c_2, c_4	0.739	0.6045	0.6691
c_2, c_6	0.406	0.3573	0.3178
c_4, c_6	0.440	0.3324	0.3084

Table 5.3 Comparison of Correlation Coefficient Estimation

It is also of interest to compare the Bayesian inversion result with the results obtained from all the previous approaches. By comparing Figure 5.10 with Figure 3.2, it is easily seen that the optimal values of the identifiable parameters c_1, c_2, c_4 and c_6 from the conjugate-gradient optimization algorithm are nearly the same as the maximum likelihood estimations from the Bayesian approach, nevertheless, the Bayesian approach gives a more complete description of the inversion results in that it also includes the inverse uncertainty as part of the information. For the non-informative parameters c_3, c_5, c_7 with the conjugate-gradient method, the MCMC approach yields widely spreading profiles across their pre-specified physical bounds and also the information that the parameters c_3, c_5 have higher probability of clustering at the upper bound of their physical ranges. The comparison between Figure 5.10 and Figure 4.2 yields the similar observation. Through comparison, we see that the Bayesian inversion should be a preferred approach for inverse analysis, as it gives the whole picture of the probability density function of the parameters from which one can "see the trees as well as the forest". Moreover, we can estimate all the interested quantities that summarize the inversion results such as maximum likelihood values, mean values, variances and correlations etc.

5.8 Data comparison

Now we show the comparison between the simulated data sets and the observed data sets. To obtain the comparison, we fixed values for c_1, c_2, c_4, c_6 at

$$1.8242\times10^{-3},\ 1.0669\times10^{-4},\ 0.9360\times10^{-3},\ 1.2729\times10^{-4}$$

and randomly generated a set of numbers for c_3 and c_5 according to the modes indicated by Figure 5.10: $c_3 \in [0.02, 0.027], c_5 \in [5.5\times10^{-3}, 8.5\times10^{-3}]$; while c_7 was randomly generated according to uniform distribution over its physical bound. The fittings are in general satisfactory with R^2 being high, as are indicated by Figure 5.11.

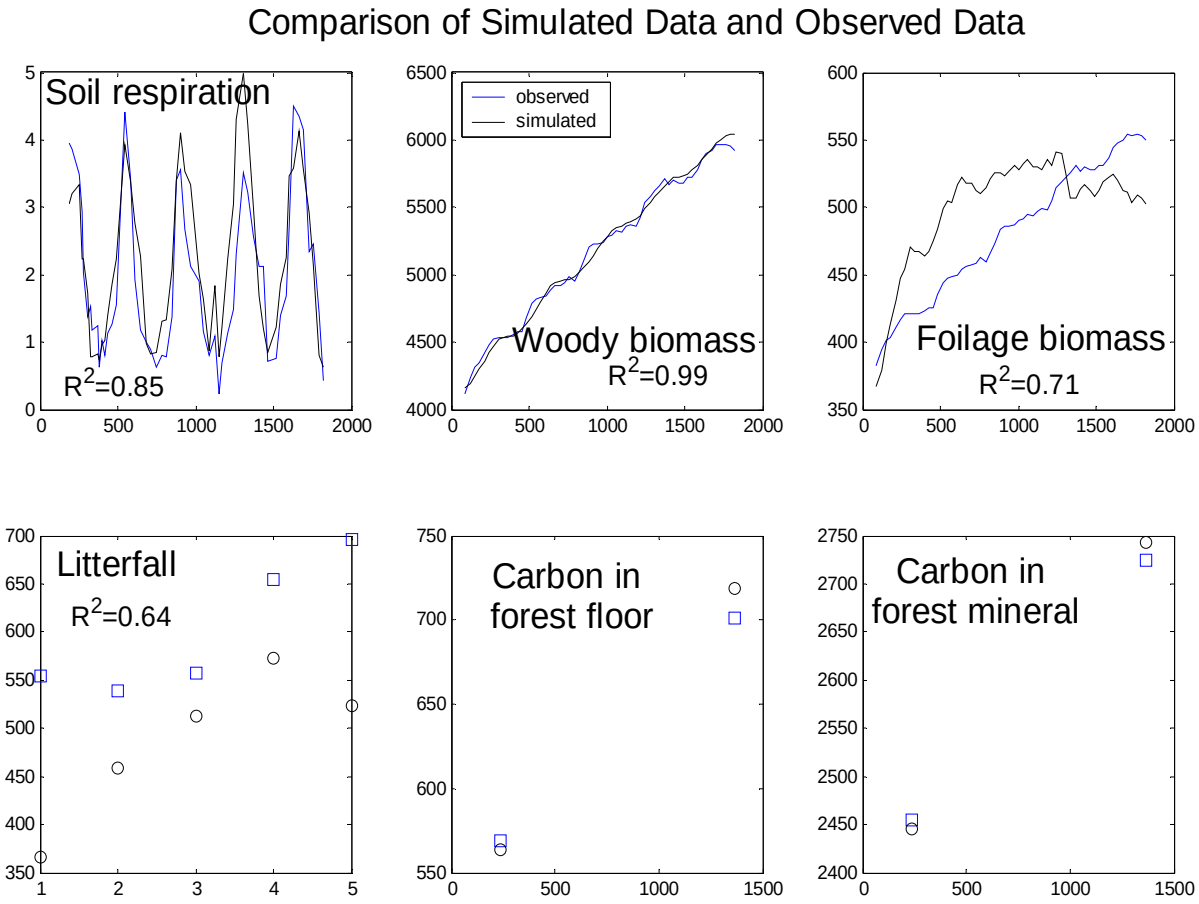

Figure 5.11 Data comparison. The figure shows the six observational data sets used in the inverse analysis and the six simulated data sets based on the inversion result.

5.9 Model prediction

Now we construct the posterior predictive distribution of the pool sizes based on the above simulation result. The predictive probability distribution of $\hat{Z}$ given Z is defined by

$$p(\hat{Z} \mid Z) = \int p(\hat{Z} \mid \theta)\, p(\theta \mid Z) d\theta$$

To actually construct the predictive distribution, samples from the above MCMC simulations were fed to the model equation (2.1). The properties of the output were summarized by the following marginal distributions and cumulative distributions.

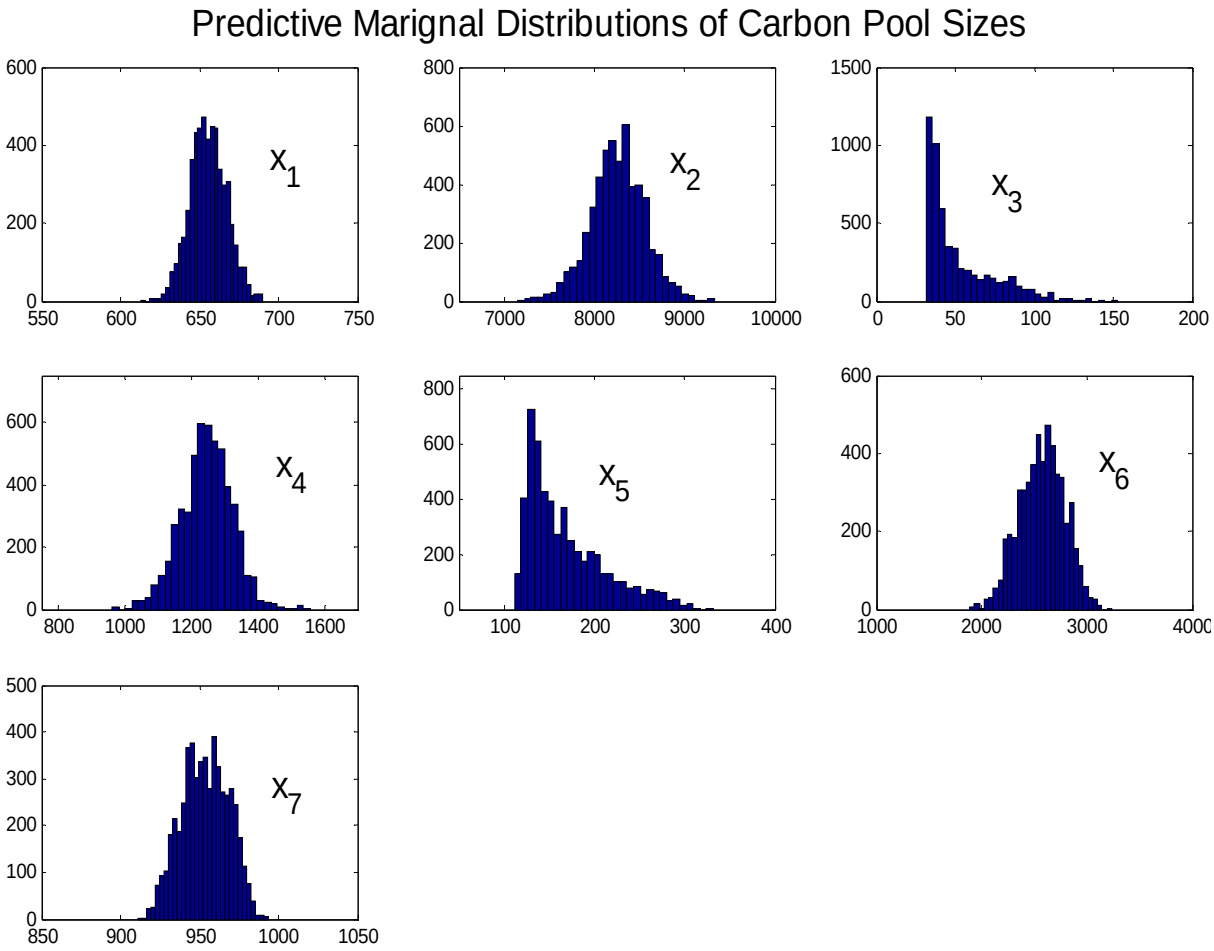

Figure 5.12 Histograms (un-normalized predictive marginal distributions) of carbon pool sizes predicted in the year of 2010.

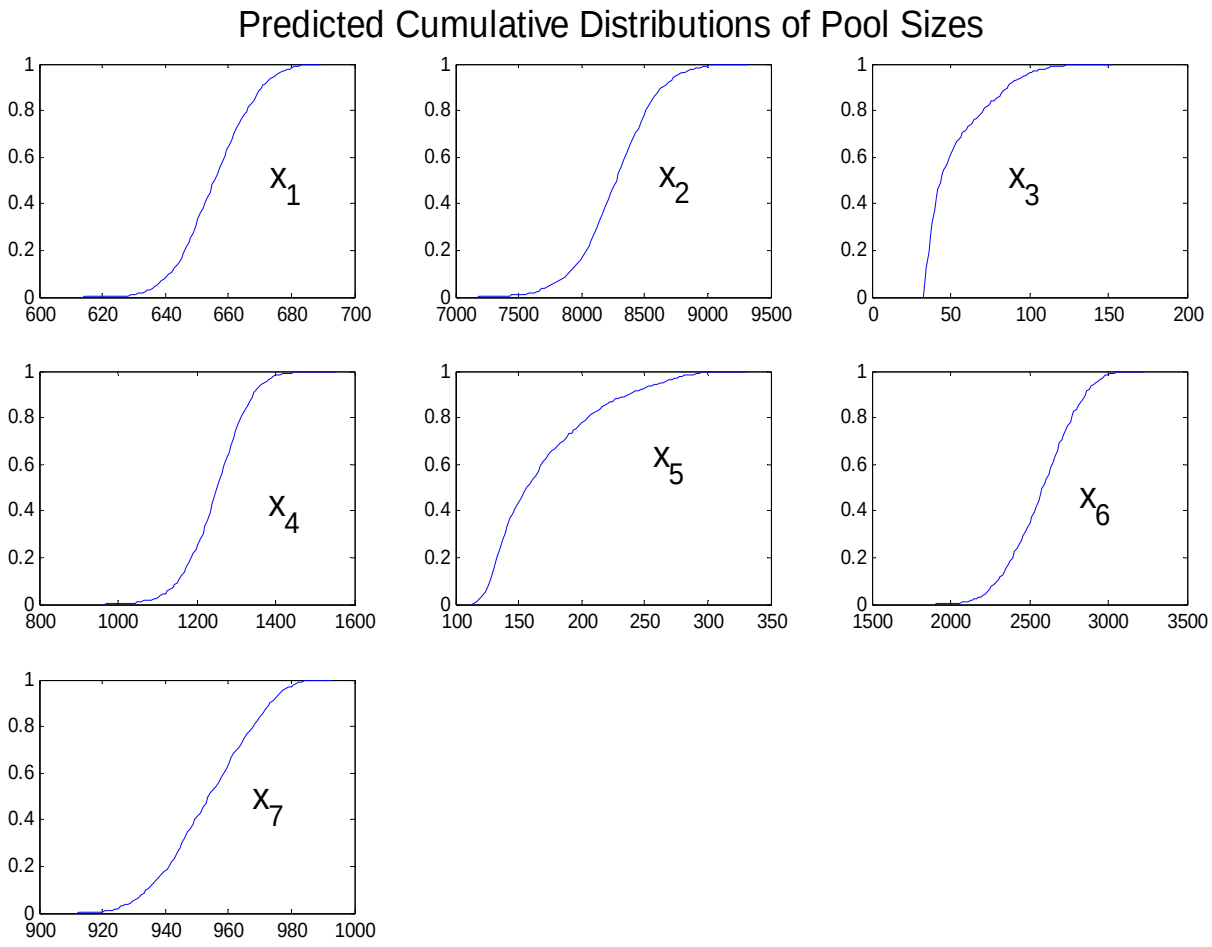

Figure 5.13 Predictive cumulative distributions of carbon pool sizes in the year of 2010.

CHAPTER 6

CONCLUSIONS AND FURTHER WORK

In this thesis, we studied the inverse problem of a general type of biogeochemical cycles that describes the carbon sequestration mechanism of terrestrial ecosystem from both deterministic and probabilistic approaches. For the deterministic approaches, we developed the Jacobian and Hessian matrix of the cost functional under a general model structure setting and multiple data sets constraints. We then proposed two descent-type algorithms, applied them to the practical model, and made estimates for the model parameters. By tracing the searching paths, we revealed the identifiability/non-identifiability of the parameters. In order to include the situations where the functions in the model were not smooth and the possible existence of multi-minima, we introduced the stochastic search algorithms GA and SA. It was shown that the numerical testing results were the same even under non-smooth functions in the model. However, these

approaches have limitations in that they only give point estimations and could not reveal the uncertainty in the inversion caused by observational uncertainties.

To include the inverse uncertainty study, we introduced the Bayesian paradigm. By specifically assuming the Gaussian type of random errors in the observational data sets, we proposed the posterior probability density function (PPDF) for the model parameters. Two approaches were developed to study the PPDF – the Laplace approximation approach which approximated the PPDF with Gaussian function based on the Hessian matrix information, and the MCMC simulation approach which sampled the PPDF directly and analyzed the samples to obtain the PPDF properties. The first approach only works in the situation where the PPDF has a single distinguished peak or is nearly quadratic in model parameters. The second approach overcomes this limitation, and is generally applicable to PPDF of any shape due to the fact the MCMC approach regards the PPDF as a stationary distribution of a Markov chain. However, standard MCMC algorithms are computationally costly, especially when applied to the biogeochemical model where each simulation involves the solution of the forward differential equation over a significant time span. To increase the efficiency of the MCMC simulation, we combined the Hessian matrix with the proposal probability in the Metropolis-Hastings algorithm. Application to the practical model showed the number of samples accepted was considerably increased.

Through the comparison, we see that, for inverse problems, the probabilistic approach not only gives the parameter values but also the description of inverse uncertainty, and is a more complete approach than the non-probabilistic approaches.

Though this thesis gives a relatively complete study of the inverse analysis methods that could be applied to the ecosystem model, there are still further issues to be addressed:

1. We did not address the convergence aspects of the MCMC simulation, nor did we delve deep to the theoretical aspects of the algorithms. One reason for this negligence was perhaps due to the fact that the study was strongly application motivated; the other reason was perhaps because the convergence was fairly straightforward for the practical numerical model we used – all approaches showed that the cost function or the PPDF had only one single well-defined peak for the identifiable parameters, and it was obviously not difficult for MCMC to find the single peak. However, there will be situations where the model space and data space together define a cost function or PPDF in a complex manner in which cases the convergence analysis will become very important for MCMC to reliably reveal the PPDF structure. There is a very rich literature about the convergence analysis for MCMC that needs to be introduced with further study (see Cowles & Carlin (1996) for an excellent review of Markov Chain Monte Carlo Convergence Diagnostics).
2. We studied the uncertainty caused by measurement uncertainty. Uncertainty caused by model structure was not addressed. One interesting question is: based on the current information available, what the model structure should be. This means, instead of fixing the number of pools in the model, we let the number of pools be decided by the current information available. This could avoid the over-

parameterization of model and enable the information to be concentrated on the key parameters.

REFERENCES

1. Aarts, E., Korst, J. Simulated Annealing and Boltzmann Machines, John Wiley & Sons, 1989.
2. Box & Tiao, Bayesian Inference in Statistical Analysis, Published by Addison-Wesley, 1973.
3. Comins, H. N., R. E. McMurtrie, Long-term biotic response of nutrient-limited forest ecosystems to co2 enrichment: equilibrium-behavior of integrated plant-soil models, Ecological Applications 3:666 -681, 1993.
4. Cowles, M.K. and Carlin, B., Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91: 883-904, 1996.
5. De Jong, K. A., An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Thesis, University of Michigan, Ann Arbor, MI., 1975.
6. Dowsland, K.A., Simulated Annealing. In Modern Heuristic Techniques for Combinatorial Problems (ed. Reeves, C.R.), McGraw-Hill, 1995.

7. Enting, I.G., Inverse Problems in Atmospheric Constituent Transport, Cambridge Atmospheric and Space Science Series, 2002.
8. Fitzpatrick, B.G., Bayesian Analysis in Inverse Problems, Inverse Problems (7), 675-702, 1991.
9. Gelfand A. E., Smith, A. F. M. Sampling based approaches to calculating marginal densities. Journal of the American Statistical Association, 85:398--409, 13, 1990.
10. Geman S. and Geman D., Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6:721--741, 1984
11. Gamerman, D., MCMC: Stochastic simulation for Bayesian inference, Chapman & Hall/CRC, New York, 1997.
12. Gill, J. Bayesian Methods – a Social and Behavioral Approach, Published by Chapman & Hall/CRC.
13. Goldberg, D.E., Genetic Algorithms in Search, Optimization and machine learning, Published by Addison-Wesley Pub Co., 1989.
14. Goldberg, D. and Vosser, S., "Optimizing Global-Local Search Hybrids" Proc. the Genetic and Evolutionary Computation Conference, pp 220-228, Morgan Kaufmann, 1999.
15. Grefenstette, J. J., Optimization of Control Parameters for Genetic Algorithms, IEEE Trans. Syst., Man, Cyber. SMC-16, 122-128, 1986.
16. Hadamard, J., Lectures on Cauchy's Problem in Linear Partial Differential Equations, New Haven, CT, Yale University Press, 1923.

17. Hansen, P. C., The L-curve and its use in the numerical treatment of inverse problems, http://www.sintef.no/static/AM/vskoler/.
18. Hastings, W.K., Monte Carlo sampling methods using Markov chain and their applications, Biometrika, 57, 97-109, 1970.
19. Haupt, R.L., Haupt, S.E., Practical Genetic Algorithm, Published by Wiley Interscience, 1998.
20. Hensel, Edward, Inverse Theory and Applications for Engineers, Prentice-Hall, 1991.
21. Holland, John H., Genetic Algorithms, Scientific American, 1992.
22. Kirkpatrick, S , Gelatt, C.D., Vecchi, M.P. 1983. Optimization by Simulated Annealing. Science, vol 220, No. 4598, pp671-680
23. Lundy, M., Mees, A. 1986. Convergence of an Annealing Algorithm. Math. Prog., 34, 111-124
24. Luo, Y., Luther W. White, Josep G. Canadell, Evan H. DeLucia, David S. Ellsworth, Adrien Finzi, John Lichter, and William H. Schlesinger Sustainability of terrestrial carbon sequestration: A case study in Duke Forest with inversion approach, Global Biogeochemical Cycles, Vol. 17, No.1, 2003.
25. Luo, Y., and J.F. Reynolds, Validity of extrapolating field CO2 experiments to predict carbon sequestration in natural ecosystems, Ecology 80:1568-1583, 1999.
26. Mellillo, J.M., A.D. McGuire, D.W. Kicklighter, B. Moore III , C.J. Vorosmarty , and A. I Schloss, Global climate change and terrestrial net primary production, Nature 363 :234- 240, 1993.

27. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. 1953. Equation of State Calculation by Fast Computing Machines. J. of Chem. Phys., 21, 1087-1091.

28. Mitra, D., Romeo, F., Sangiovanni-Vincentelli, A. 1986. Convergence and Finite Time Behavior of Simulated Annealing. Advances in Applied Probability, vol 18, pp 747-771.

29. Nicholls, G.K., Bayesian inference and Markov chain Monte Carlo by example, 2004, available on line at http://www.math.auckland.ac.nz/~nicholls/linkfiles/papers/NichollsKuopio04.pdf

30. Parton, W.J., D.S. Schimel , C.V. Cole, and D.S. Ojima, Analysis of factors controlling soil organic matter levels in Great Plains grasslands. Soil Science Society of America Journal 51:1173 – 1179, 1987.

31. Press, W.H., Teukolsky SA, Vetterling WT, Flannery BP, Numerical Recipes IN Fortran 77: the Art of Science Computing. Cambridge University Press, UK, 1992.

32. Rabenstein, A.L., Introduction to Ordinary Differential Equations, Academic Press, 1972.

33. Rastetter, E.B., G.I. Agren and G.R. Shaver, Responses of N-limited ecosystems to increased CO2: a balanced-nutrition, coupled-element-cycles model, Ecological Applications 7:444 – 460, 1997.

34. Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. 1996. Modern Heuristic Search Methods. John Wiley & Sons.

35. Russell, S., Norvig, P. 1995. Artificial Intelligence A Modern Approach. Prentice-Hall.

36. Roberts, G.O. and Rosenthal, J.S., Markov chain Monte Carlo: Some practical implications of theoretical results, Canadian J. Stat. 26, 5-31, 1998.
37. Spall, J. C., Introduction to stochastic search and optimization, , simulation and control, Published by Wiley-Interscience, 2003.
38. Spall, J. C, Estimation via Markov Chain Monte Carlo, Proceeding sof the American Control Conference, Anchorage, AK, May 8-10, 2002.
39. Tan, S.M, C. Fox and G.K. Nicholls, Inverse Problems, 2004, available on website http://www.math.auckland.ac.nz/~nicholls/707/.
40. Tarantola, Albert. Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation, Elsevier Science Press, Amsterdam, 1987.
41. Thompson, M.V, and J.T. Randerson, Impulse response functions of terrestrial carbon cycle models: method and application, Global Change Biology 5: 371-394, 1999.
42. Tikhonov, A., On the solution of incorrectly stated problems and method of regularization. Dokl Akad Nauk SSSR, 151:501, 1963.
43. Von Mises. Mathematical Theory of Probability and Statistics. Academic Press, New York, 1964.
44. Vose. M. D., the Simple Genetic Algorithm: Foundations and Theory (Complex Adaptive Systems), MIT Press, 1999.
45. White, L., Luo, Y., Estimation of carbon transfer coefficients using Duke Forest free-air CO2 enrichment data, Applied Mathematics and Computation, 130, 101-120, 2002.