Application of Inverse Problem Theory and Markov Chain Monte Carlo Technique

Vinay Krishna Gade

A dissertation submitted for the partial fulfilment of
BS-MS dual degree in Science

Indian Institute of Science Education and Research Mohali
April 2013

Certificate of Examination

This is to certify that the dissertation titled "Application of Inverse Problem Theory and Markov Chain Monte Carlo Technique" submitted by Mr. Vinay Krishna Gade (Reg. No. MS08020) for the partial fulfillment of BS-MS dual degree programme of the Institute, has been examined by the thesis committee duly appointed by the Institute. The committee finds the work done by the candidate satisfactory and recommends that the report be accepted.

Dr. Lingaraj Sahu Prof. Sudeshna Sinha Prof. Somdatta Sinha
(Supervisor)

Dated: April 26^{th}, 2013

Declaration

The work presented in this dissertation has been carried out by me under the guidance of Prof. Somdatta Sinha at the Indian Institute of Science Education and Research, Mohali.

This work has not been submitted in part or in full for a degree, a diploma, or a fellowship to any other university or institute. Whenever contributions of others are involved, every effort is made to indicate this clearly, with due acknowledgement of collaborative research and discussions. This thesis is a bonafide record of original work done by me and all sources listed within have been detailed in the bibliography.

Vinay Krishna Gade
Dated: April 26^{th}, 2013

In my capacity as the supervisor of the candidate's project work, I certify that the above statements by the candidate are true to the best of my knowledge.

Prof. Somdatta Sinha
(Supervisor)

Acknowledgment

First, I would like to thank my project guide for giving me such wonderful opportunity to learn. She was very patient and kind enough to help me and constantly pushing my limits to do my best. During this one year of thesis I not only improved my knowledge by bounds but also learned the art of scientific research from her. This has been a great experience for me and I hope it will be really helpful for my future. Next, I would like to thank Dr. Amit Apte, TIFR-CAM who has been constantly guiding me through out this project. He taught me Inverse Problem theory and MCMC techniques which is the center part of my project. I learnt a great deal of things from him especially some good techniques in scientific programming. I am infinitely grateful for him for going through my long messy programs and debugging them, which is really a difficult job. He is my role model when it comes to way of learning things. Thank you once again, Amit sir. I would also like to thank my colleague Mr. Ashutosh Srivastava, who gave regular inputs about current research and interesting problems, which helped a great deal in keeping my enthusiasm. And finally I would like to thank the IISER community at large especially the IISER, Library and computation facility. Thank you.

Vinay Krishna Gade

List of Figures

Contents

Abstract

Determining the model space parameters from given data is an important research area in Mathematics and Statistics having applications in Physics, Chemistry, Biology and Engineering. In this thesis we consider having data on the observable variables for a certain model having multiple parameters, and attempt to do parameter estimation for the model parameters using the data and some given information. We proceed to solve the problem treating it as an inverse problem, and build the probability density functions for the model and data parameters. The probability functions we have constructed using the system equations, can't be fully determined due to mathematical complexity. So, we use the Markov Chain Monte Carlo (MCMC) method for estimating the model parameters. We do various kinds of analysis of the estimated parameters by substituting back into the system equations, trying to verify whether they are consistent with the model or not. In this thesis, first we describe the model space and data space, give a basic introduction to some probability and inverse theory, and then discuss the MCMC approach. Finally, we apply the concepts and methods of inverse problem theory and MCMC techniques on an example problem called Ellipse model (with two parameters), and give detailed analysis of the parameter estimation procedure. We then consider the Lorenz model, which is known to display a variety of dynamics, stable to chaotic, for different values of its three parameters, and give a brief description of how to do it's parameter estimation.

Chapter 1

Introduction

1.1 Inverse Problem Theory

1.1.1 Data and Model Spaces

Let us consider some physical system or process. To describe this system mathematically, we need to define the properties and the relations between these properties using mathematical expressions or equations. This is termed as the "model" for the process under study. These model system equations are derived from the initial knowledge we have about the process/system. Since this knowledge may be incomplete, especially in case of complex systems, these relations could be explaining the actual system only to an extent and not completely. They have uncertainties, which need to be determined. To verify the model, we do experiments and collect some data of the observables describing the properties of the physical process/system. For example, to measure the gravity of earth we use a device, which can measure the distance traveled by the body. If the device is attached to a vertically falling body (with a certain initial velocity, u), then measurements of distance can be taken, using the device, for n times at $\{t_1, t_2, t_3, \dots t_n\}$. We also have a initial condition that initial velocity is u. So, the data collected is $\{s_1, s_2, s_3, \dots s_n\}$ at times $\{t_1, t_2, t_3, \dots t_n\}$. The system equation for this process is known to be, $s = ut + gt^2/2$, where g is the gravitational constant. We can calculate the value of g from this simple expression if we know s at different t. On the other hand, we can substitute the data we have, and see what the g value comes out to be for the given data set. The analysis, how the g is determined will be discussed later.

Let us classify the various parameters we have. The set $D = \{s_1, s_2, s_3, \dots s_n\}$ is the

data collected, so these are called the 'data parameters' in the 'data space'. The set $\{t_1, t_2, t_3, \ldots t_n\}$ is taken as the 'prior information' which is bound to be known well or measured with good certainty. We also need to determine the initial velocity u and gravitational constant g values. These are called the 'model parameters' in the model space $M = \{u, g\}$. Hence, 'data space' or 'data parameters' consists of the experimental data we have with us after performing the experiment. Where as the model parameters are the parameters in the system equations which define the properties of the physical system or a process. But it is important to note that most physical systems have several properties which are observable, and as the observations change the 'data space' also changes. This will result in re-parametrization of the system equations with a new 'model space'. So, model space is akin to conventional type of space, since it depends on what properties of the physical system we choose to observe. Usually, of all the possible parametrizations the minimal set of model parameters is chosen. This reduces the mathematical rigor of the inverse problem. And, also any two parametrization are equivalent if they are bijective. This description can be put in more abstract manner, taking every set of values for model parameters as a point in the n-dimensional parameter space, where n is the number of parameters. Here each point represents a model, that may or may not fit the system equations, for a given set of data.

1.1.2 Forward Problem

A simple definition of a forward problem would be, to predict the values of the observable parameters D, by constructing the system equations. So, the major part of a forward problem would consist of formulating the system equations. These equations are constructed based on the earlier knowledge that we possess about the problem. So, the steps to describe the forward problem for a physical system or process would be to identify the data and model space first, and then proceed to formulate a theory and state the postulations, which would be the system equations. So, mathematically, if the $d \in D$ Data space, then we try to derive a function $g(m) = d$, where $m \in M$, the Model space. To put it precisely, $g(m)$ consists of the vector m and the 'prior information' as variables. Therefore, deriving the mathematical equations from the postulations/assumptions and from the previous knowledge we had about the problem is involved in solving a Forward Problem.

1.1.3 Inverse Problem

The Inverse problem is about the model parameter estimation. In this case we are given the data, which are experimental observations, and also possess some knowledge about the theory and postulations, i.e the system equations representing the system. What we need to do is estimate the model parameters that best fit the data. In the example described previously in 1.1.1, we defined the model and data space. The data space is the experimental observations, and we also have the system equations. Based on the two we try to derive the model parameters from the data space. The final set of estimations we get for the model space is the solution for the inverse problem. In the next section, we describe a set of tools required to do the above, and discuss about how to derive a solution for an Inverse Problem.

1.1.4 Solution of an Inverse Problem

The Data and Model space (or manifold) are determined and we also have derived the system equations. A joint manifold is constructed from these two manifolds,say W. Such a manifold is useful when we try to do a probabilistic interpretation of the data and model space. We construct the probability density functions in the space of $W = D \times M$, which is the cross product of the two spaces. If $d = (d_1, d_2, d_3, \ldots d_r) \in D$ and $m = (m_1, m_2, m_3, \ldots m_s) \in M$, then $w = (d_1, d_2, \ldots d_r, m_1, m_2, \ldots m_s) \in W$. We can now build a joint probability density function for the manifolds D and M. Once, the parameters and spaces are determined, all we need to do is do a probabilistic interpretation for them. To proceed from here, we suppose that the forward problem is not completely true and the system equations are not deterministic and thereby could be given an estimation as some probability. If we introduce some kind of uncertainty in the system equation $d = g(m)$, say ζ, then the system equations would be

$$d = g(m) + \zeta \tag{1.1}$$

ζ is the uncertainty in the system equations, therefore –

$$d - g(m) = \zeta \tag{1.2}$$

So, the above difference is the uncertainty in the system equations. What one can do now is to assign a distribution for the uncertainty. This implies that the probability that we have for a given amount of uncertainty, is now given by a function over ζ.

This distribution could be Guassian or any other kind, suitable for the purpose. This tells us that, for a given point in the model space, how valid is a data point in the data space. This is equivalent to putting a conditional probability for a point in data space for a given model space point. Next we define a joint probability density, which describes the physical postulations about the system and also explains the uncertainties in the mathematical modeling.This joint probability density, say $\Theta(d, m)$, is called the theoretical probability density, which is given as –

$$\Theta(d, m) = \theta(d|m) \ \mu_M(m) \tag{1.3}$$

where $\theta(d|m)$ is the conditional probability as described above and $\mu_M(m)$ is a marginal homogeneous probability density function in the model space as given below –

$$\mu_M(m) = \int_D \mu(d, m) \ \mathrm{d}d \tag{1.4}$$

where the $\mu(d, m)$ is the joint homogeneous probability distribution over the joint manifold W. Constructing and calculating the theoretical probability density solves most of the problem. The $\theta(d|m)$ in the theoretical probability signifies the uncertainty in the system equations i.e how good is the data space for a given point in model space. Therefore, $\theta(d|m)$ is a distribution for uncertainty in the system equations, hence

$$\theta(d|m) = \delta(\zeta). \tag{1.5}$$

But, what we should not neglect is the initial idea or estimation about the both data and model parameters. This information, which we possess before conducting the experiment or trivial things that can be inferred from the data, is called 'A Priori' information. A priori information on data parameters is basically dependent on the efficiency of instrument in calculating the observables. We can interpret them as the error analysis on the data, i.e., errors caused due to inaccuracies in the instrument measurements. Where as for the model parameters, we build a distribution which could be a possible estimation for the model parameter values. This is usually done using the data or some kind of intuitive guess. And also, the observable errors are large or small depend relatively on the modelizaton errors.

Let $\rho_M(m)$ and $\rho_D(d)$ represent the A priori information over the model and data manifolds respectively. Assuming these both states of information are independent

from each other, we can give the joint prior information as –

$$\rho(d,m) = \rho_D(d)\ \ \rho_M(m) \tag{1.6}$$

Once the theoretical and joint A priori probability densities are calculated all we need to do is take conjunction of them, which will result in the posterior probability density, given as –

$$\sigma(d,m) = \kappa\ \ \frac{\rho(d,m)\ \ \Theta(d,m)}{\mu(d,m)} \tag{1.7}$$

Done that, now we have to determine a probability distribution, which explains how well a model point fits the data point, which is the set of observables we have at hand. Such a function is called Likelihood function, denoted as $L(m)$. Likelihood function is the solution of the inverse problem, which gives a probabilistic interpretation how the model points are distributed for given data. Therefore, Likelihood function is directly proportional to the posterior information we have in the model space. Therefore –

$$\sigma_M(m) = \kappa\ \rho_M(m)\ L(m) \tag{1.8}$$

$$\text{where, } L(m) = \int_D \frac{\rho_D(d)\ \theta(d|m)}{\mu_D(d)}\ \mathrm{d}d \tag{1.9}$$

is the solution of the inverse problem. Mathematically the above posterior information over the model space represents the solution of the inverse problem but it is not really helpful until we determine the normalization constant κ, which is mathematically a rigorous derivation especially when the dimension of the model space is large. So, we use this function in a different manner by calculating its value at random samples of the model space. Hence, we need a methodology to estimate the model parameters by randomly sampling the model space. For this same purpose we are going to introduce the technique Monte Carlo Markov chain (MCMC). For the simulations in the MCMC we will be using much lighter version of the likelihood function. It is taken as directly proportional to the product of the prior on model space and the probability distribution over the errors in modelization which is $\theta(d|m)$. This much simpler function will serve the purpose or not will depend on the nature of the problem whether it accepts the simplification and gives solutions nearer to full rigorous models. Therefore, the Likelihood function we are going to use is given as,

$$L(m) = \theta(d|m)\ \rho_M(m) \tag{1.10}$$

Once, the Likelihood function is constructed, we are ready to run the MCMC.

1.2 Markov Chain Monte Carlo (MCMC)

1.2.1 Markov Chain

A Markov chain can be described as a stochastic process such that the future states are independent of the past states given the present state. Mathematically, if we have a sequence of independent and identically distributed ($i.i.d$) random variables $\{X_i\}_{i\geq 0}$ which take values from a state space $\{a_i\}_{i\geq 0}$ such that

$$P(X_{i+1} = a_{i+1}|X_i = a_i, X_{i-1} = a_{i-1}, \ldots, X_0 = a_0) = P(X_{i+1} = a_{i+1}|X_i = a_i) \tag{1.11}$$

then we say that, $\{X_i\}_{i\geq 0}$ is a Markov chain. Here, we notice that the process is a discrete time process. Most of the real data experiments using markov chains are discrete time step process. The state space can be continuous or discrete depending on the model space we are using. We will be working on *discrete-time discrete-state* processes, so any markov chain mentioned in future is a discrete step process unless pointed out specifically. Now, where does markov chains leads us to. Before doing that let us define a *transition matrix* M for a state space. It is a matrix which consists the transition probabilities of going from any given state to every other state. So, if $\{a_i\}_{i\geq 0}$ is the state space from which X_i's take values, then any element $m_{kl} \in M$ is the transition probability of going from state a_l to a_k. Therefore, for a discrete state space of size n,

$$M = \begin{pmatrix} m_{00} & m_{01} & \ldots & m_{0n-1} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n-10} & m_{n-11} & \ldots & m_{n-1n-1} \end{pmatrix} \quad \text{where,} \quad m_{kl} = P(k \to l). \tag{1.12}$$

These probabilities in the transition matrix are calculated using various factors in the physical system that influence transition between states. From an inverse problem point of view, these states represent the model state space and the solution we are trying to acquire is a distribution for these states. So, here is how we derive such

a distribution using a Markov chain. We take some initial distribution for the state space, say $\pi^{(0)}$. Therefore,

$$\pi^{(0)} = (\pi_0^{(0)} \pi_1^{(0)} \dots \pi_{n-1}^{(0)}) \tag{1.13}$$

Since the conditional distribution of X_{i+1} given $X_n, X_{n-1}, \dots, X_0$ depends only on X_n, in the next time step using the transition matrix we try to calculate the distribution $\pi^{(1)}$, which is given as –

$$\begin{aligned} \pi^{(1)} &= \pi^{(0)} M \quad \text{and} && (1.14) \\ \pi^{(k)} &= \pi^{(k-1)} M \quad \text{therefore,} && (1.15) \\ \pi^{(k)} &= \pi^{(0)} M^k && (1.16) \end{aligned}$$

Now the above equation for larger k, we will need $\pi^{(k)}$ to converge to some distribution which is independent of the initial distribution $\pi^{(0)}$. Such that,

$$\pi = \pi \, M \tag{1.17}$$

This would happen if a certain set of conditions are put on the state space. There are three such conditions and they go as follows –

- Aperiodic - If $X_i = a \in A$ then there exists no k, such that $X_{i+k} = a$, then we say state a is aperiodic. If $\forall a \in A$ are aperiodic, then A is aperiodic.
- Irreducible - If all states in the state space A communicate i.e $m_{ij} > 0 \ \forall \, i, j < n$ then we say A is irreducible.
- Positive Recurrent - Let us define for any $a \in A$, $T_a = inf \ \{i : X_i = a, i \geq 1 \mid X_0 = a\}$. Now, a state a is recurrent if $P_a(T_a < \infty) = 1$, where P_a is a probability distribution over $\{X_i\}_{i \geq 0}$ with $X_0 = a$. A recurrent state is called positive recurrent if $E_a(T_a) < \infty$, where E_a is the expectation value with respect to the probability distribution P_a.

1.2.2 Monte Carlo Methods

In this section we would give brief introduction to monte carlo methods and how this numerical technique is useful for practical purposes and brings simplicity to complex problems. As described earlier, to evaluate the likelihood function one needs to determine the normalization constant but since the function is higher dimensional, integrating it would be a rigorous problem. To bypass such a situation we adopt monte carlo methods. Monte carlo methods involves randomly sampling the state space and calculating the function value at that point and sum it. This would give us an ample idea what would be the integrated value of the function, even though it is a mere estimation. Anyways, in the present problem we are dealing with we would not try integrate the likelihood function we would use something similar i.e random sampling. We try to calculate the values of the likelihood function at different points in the model space, if the state space is aperiodic, irreducible and positive recurrent then assuming the sequence to be a markov chain should converge to a stationary distribution. The process of this will be explained more clearly in the algorithm for MCMC. Let us see how are we using the monte carlo methods in distribution. Now suppose $\{X_i\}_{i=0}^{n}$ is a sequence of *i.i.d* random variables such that they converge to an *i.i.d* X in distribution. Point to note here is, the sequence of random variables are technically simulated for the purpose, which will converge to a stationary distribution. Define,

$$\hat{\mu} = \frac{1}{n+1} \sum_{i=0}^{n} X_i \tag{1.18}$$

If X has a mean μ and standard deviation σ, then by Central limit theorem we have the sample mean $\hat{\mu}$ converges to μ and the estimated variance given as,

$$\hat{\sigma}^2 = \frac{1}{n+1} \sum_{i=0}^{n} (X_i - \hat{\mu})^2 \tag{1.19}$$

converges to the σ. So in our problem we are going to use monte carlo to simulate random variables which converge to the stationary distribution we need.

1.2.3 The algorithm for MCMC - Metropolis Hastings

The Metropolis Hastings algorithm is quite simple. Before going there let me define a function called *proposal density function*. The proposal density function is where the samples from state space for the simulation are picked up. The choice of this function is purely dependent on the problem and how much information we have on the model parameters. For this same purpose we need to have an initial idea how the model parameter values look like, this would increase the efficiency of the simulation highly. Let us see how they really work. A proposal density function q has two arguments in the function, the present state a_0 and a newly generated state a. Now, it is constructed such that, either $q(a|a_0)$ or $q(a_0|a)$ is greater depending on which of between a and a_0 is nearer to our estimation. But such a construction is quite difficult to do since these functions are not usually normal distributions which makes it difficult to understand them. If $q(a|a_0)$ and $q(a_0|a)$ are not equal then we say it is an asymmetric type of proposal density else symmetric. Metropolis Hastings algorithm would be using asymmetric type of proposal density function. The algorithm for running an MCMC simulation would be like this –

- Step 1 - Start with some initial seed $x_{old} = x_0$ and calculate $L_{old} = L(x_0)$.
- Step 2 - Generate a new state x.
- Step 3 - Calculate the *Hastings ratio* $r = \frac{L(x)\ q(x|x_0)}{L_{old}\ q(x_0|x)}$.
- Step 4 - Generate a random number u between 0.0 and 1.0.
- Step 5 - If $(u \leq r)$
  put $x_{old} = x$ and $L_{old} = L(x)$.
- Step 6 - Goto Step 2.

So the above sequence of steps is a simple representation of the Metropolis Hastings algorithm. This algorithm is used for running MCMC programs for all problems explained in this report. In the algorithm every state is randomly sampled from the proposal density function. As seen we start with an initial seed, then we start a loop equal to the number of time steps we want to run the simulation. Inside the loop we generate a new state and calculate the Hastings ratio between the two states. Next, we generate a random number u between 0.0 and 1.0. Then we do the ratio test, if the new state passes the test then we replace the old state variable with the new state value, similarly for the likelihood function value. After the ratio test the loop goes

back and generates a new state. During this all process we need to check whether the distributions are really converging. This is done by testing for convergence in mean of all accepted states at every time step, which will help us to note when the sequence starts converging in distribution. A point to note here is that, whenever a state is rejected then for calculating the mean the old state is used at that time step. Hence, only then the mean converges to the right values of the state space, which is the solution to our problem. Next thing we are concerned during a MCMC simulation is the acceptance rate. The acceptance rate is the percent of states that are accepted. A check for acceptance rate could be done at every time step, this helps in seeing which parts of the state space have higher probabilities. Also, a good MCMC simulation would have a overall acceptance rate of around 20 percent. Using the acceptance rate and likelihood function values at every time time step we can tweak the proposal density function so that jumps could be made accordingly if the simulation got stuck between a certain range of state space values.

Chapter 2

Results

We now apply the methods and techniques described in the earlier chapters to two specific problems whose models are known. The first one - the *Ellipse Problem* - estimates the two parameters describing an ellipse from a set of data of the two variables. We show the analysis in detail. The second one - *the Lorenz Equation* - is a model to describe fluid flow, is known to display a variety of dynamics, stable to chaotic, for different values of its three parameters. We give a brief description of how to do the estimation of its three parameters from the chaotic orbits.

2.1 The Ellipse Problem

Here we apply the inverse problem theory and the MCMC technique on the simple problem of estimating the parameters a and b of an elliptic orbit motion. So, we have n co-ordinate points $\{(x_i, y_i)\}_{i=1}^{n}$, which is the data or observables. The forward problem in this case would be determining the system of equations, which in this case is a simple parametric equation for an ellipse, given as –

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{2.1}$$

Using the above equation we can determine the value of, say x at a given y for different values of a and b. This would be the forward problem solution. For the inverse problem we generate data taking $a = 1.3$ and $b = 0.4$, then the ellipse would like as shown in Figure 2.1

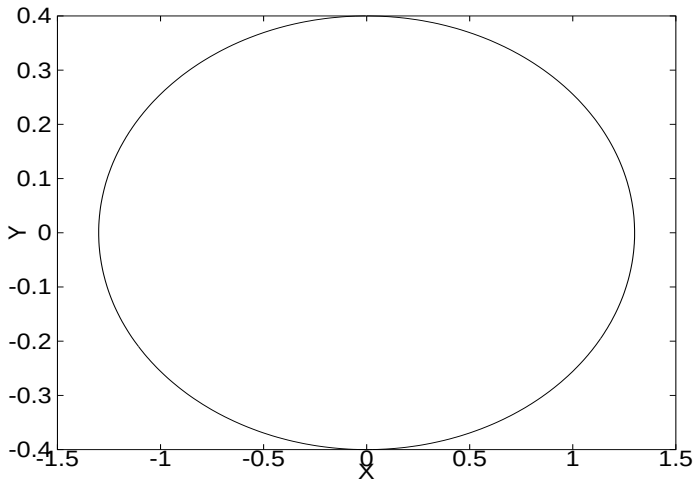

Figure 2.1: Ellipse with a=1.3 and b=0.4

During an experiment the data always has some noise or deviation from the actual values. So, the data we get through a real experiment would have noise in it. To generate some synthetic data for this problem, we pick some data points from the ellipse and add noise to the co-ordinate points. And, this data set, given in the Table A.1, would be used for the inverse problem. The data points,when plotted, is shown in Figure 2.2 –

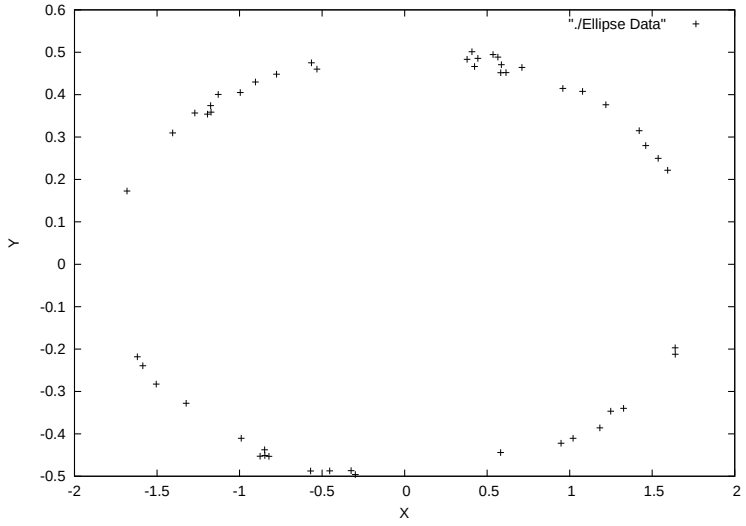

Figure 2.2: Plot of Ellipse Data

When compared to Figure 2.1, it is clear that the data exhibits stochasticity and the elliptic orbit is not as clearly defined as in Figure 2.1.

In an inverse problem we are given the data set (x and y) and the system equations, and we need to determine the values of a and b. In a forward problem, given a y, one can determine the value of x for a certain a and b. We take the $y_i's$ as *a priori* information, which means that the values of y component are deterministic values (or, with no error), and use the x component in the data set as the observables. So the data space consists of $\{x_i\}_{i=1}^{n}$ as the only data. So, the system equation $d = g(m)$ would look like –

$$x_i = \pm\, a\sqrt{1 - \frac{y_i^2}{b^2}} \tag{2.2}$$

Clearly the model space is a two dimensional space with its components represented as (a, b). Once the system equations, the data space and the model space are determined we are ready to solve the inverse problem.

First, we need to determine the Prior density distribution for the model space. This is done by observing the range of distances between any two points, which will give estimates for the a and b values. If the data set is spread evenly through the orbit motion, we can get the maxima and minima of the distances as the a and b estimates nearly to the right values. This shows that the prior density function is mostly dependent upon how good the data set is. Hence, a good data set will give good estimates for the inverse problem, and a bad data set will make bad estimation. The prior density function for the model space, taken as guassian distribution, would look like –

$$\rho(a, b) = \frac{1}{2\pi\ \sigma_a \sigma_b} exp\left(\frac{-(a-\hat{a})^2}{2\sigma_a^2} + \frac{-(b-\hat{b})^2}{2\sigma_b^2}\right). \tag{2.3}$$

where $\hat{a}$ and $\hat{b}$ are the estimates for a and b values,
σ_a and σ_b are the standard deviations.

Next, what we need to determine is the theoretical probability density function $\Theta(d, m) \propto \theta(d|m)$, which is given as –

$$\theta(d|m) = \prod_{i=1}^{n} \delta\left(|x_i| - \left|a\sqrt{1-\frac{y_i^2}{b^2}}\right|\right). \tag{2.4}$$

$$= \frac{1}{(2\pi)^{n/2}\,\sigma^n} \prod_{i=1}^{n} exp\left[-\frac{\left(|x_i| - \left|a\sqrt{1-\frac{y_i^2}{b^2}}\right|\right)^2}{2\,\sigma^2}\right] \tag{2.5}$$

where σ is the standard deviation of the data set $\{x_i\}_{i=1}^{n}$.

Once the prior and theoretical probabilities are constructed we have the likelihood function given as –

$$L(a,b) = \frac{exp\left(\frac{-(a-\hat{a})^2}{2\sigma_a^2} + \frac{-(b-\hat{b})^2}{2\sigma_b^2}\right)}{(2\pi)^{(n+2)/2}\,\sigma^n\sigma_a\sigma_b} \prod_{i=1}^{n} exp\left[-\frac{\left(|x_i| - \left|a\sqrt{1-\frac{y_i^2}{b^2}}\right|\right)^2}{2\,\sigma^2}\right] \tag{2.6}$$

Since in the algorithm we would be using the likelihood function for calculating the Hastings ratio, where the normalization constants get canceled, so normalized constants can be ignored, similarly any probability function we try using in the simulation need not be normalized like the proposal density function. The proposal density function for any MCMC simulation is manipulated such that the samples are picked up near our estimated values. But it should also not be biased at the same time by making the range of values too narrow. This would defeat the whole purpose. So, the proposal density function in the case of ellipse problem is quite simple. It is as follows –

$$Q(m|m_0) = Q(a, a_0, b, b_0) \tag{2.7}$$

$$= \frac{1}{2\pi\,\Delta_a\Delta_b} exp\left(\frac{-(a-\beta_a a_0)^2}{2\Delta_a^2} + \frac{-(b-\beta_b b_0)^2}{2\Delta_b^2}\right) \tag{2.8}$$

where Δ_a and Δ_b are standard deviations for parameters a and b in the proposal density function, β_a and β_b are some constants which give asymmetry for the function Q.

Now, determining the values of β_a,β_b,Δ_a and Δ_b is the major part of constructing the proposal density function, which influence the kind of samples that are picked up and also control the acceptance rate. Since we have used an asymmetric function for the proposal, the values of $Q(m|m_0$ and $Q(m_0|m)$ will not be equal. So for calculating the later function $Q(m_0|m)$ value the variables can be interchanged as follows –

$$Q(m_0|m) = Q(a_0, a.b_0, b) \tag{2.9}$$

This kind of construction of the proposal density function makes operating on it quite flexible. Even though more complex and accurate functions can be used but not easy to handle inside a simulation.

Now once the data set is available to us we are ready to run the simulation. The pseudo code for the simulation would look like as follows –

```
DEFINE:  double Q(double, double, double, double) { }
DEFINE:  double L(double, double) { }
START :  main() {
         Enter the initial seed a_ini = a_0 and b_ini = b_0;
         Calculate (double) L_old = L(a_0, b_0);
         for(i=0;i<rounds;i++) {
         Generate a new state a and b from Q(β_a a_ini; Δ_a) and Q(β_b b_ini; Δ_b);
         Calculate L(a, b), Q(a, a_ini, b, b_ini) and Q(a_ini, a, b_ini, b);
         Calculate the Hastings ratio r;
         Generate a random number u between 0.0 and 1.0;
         if(u ≤ r) {
         put a_ini = a,
             b_ini = b and
             L_old = L(a, b); }      // END OF IF
           }                          // END OF FOR
         }                          // END OF MAIN
```

he C program for the simulation of ellipse problem are given in the Appendix. The data available to us is plotted in Figure 2.2. The results of the MCMC analysis with this data set, and the system of equations given in Equation 2.2, for parameter

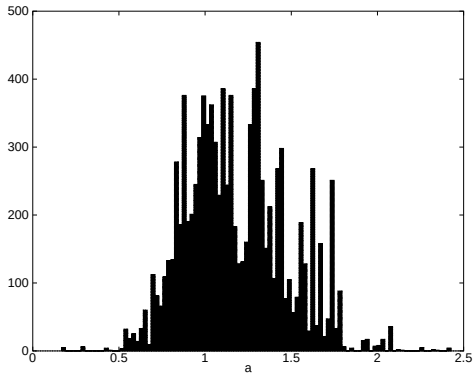

Figure 2.3: Frequency of accepted 'a' values

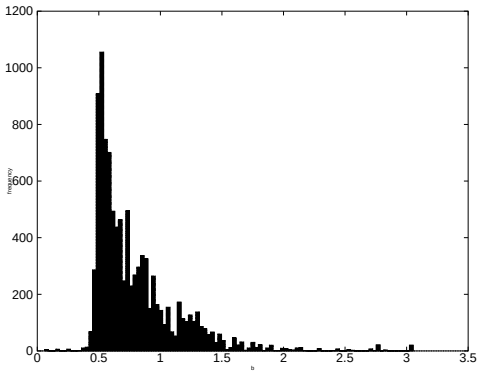

Figure 2.4: Frequency of accepted 'b' values

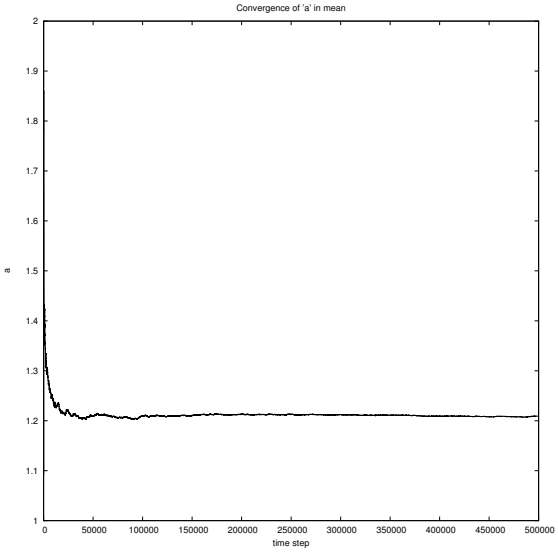

Figure 2.5: Convergence in 'a'

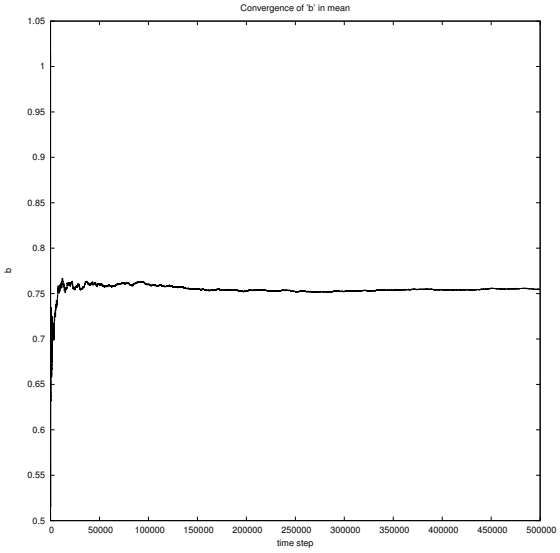

Figure 2.6: Convergence in 'b'

estimation are shown as plots. Figure 2.3 and Figure 2.4 give the distribution of accepted states for a and b. Figure 2.5 and Figure 2.6 give the corresponding estimated (converged) values of a and b on MCMC analysis. The estimated values of a and b from MCMC analysis are 1.22 and 0.77. Note that the converged values of a and b are not exactly the same as the actual values 1.3 and 0.4. This is expected as the MCMC estimation of parameters would depend strongly on the data set and the probability distributions involved.

2.2 Inverse Problem for Lorenz system

Here we give a brief mathematical description of how to construct the inverse problem for the Lorenz system of equations.
The system of equations due to Lorenz are given as follows –

$$\frac{dx}{dt} = \alpha(x-y) \tag{2.10}$$
$$\frac{dy}{dt} = x(\rho-z)-y \tag{2.11}$$
$$\frac{dz}{dt} = xy-\beta z \tag{2.12}$$

The three parameters are - α, β and ρ. Now, using the above equations we build an discrete set of equations assuming that Lorenz is a non-linear discrete system, which are given as follows –

$$X_{n+1} = \alpha(X_n - Y_n)\Delta T + X_n \tag{2.13}$$
$$Y_{n+1} = (X_n(\rho - Z_n) - Y_n)\Delta T + Y_n \tag{2.14}$$
$$Z_{n+1} = (X_n Y_n - \beta Z_n)\Delta T + Z_n \tag{2.15}$$

These above set equations can be represented as a vector. So, in $g(m) = d$ the g is a vector containing three components - X_{n+1}, Y_{n+1} and Z_{n+1}, which we will represent them as g_1,g_2 and g_3.
Clearly the model space $M = \alpha, \beta, \rho, X_0, Y_0, Z_0$, which includes the initial conditions and the data space will be the (x, y, z) co-ordinates, therefore $D = \{(x_i, y_i, x_i)\}_{i=1}^{n}$.
Once we have constructed the system equations as a vector $g(\vec{m}) = (g_1(m), g_2(m), g_3(m))$, we define a matrix S as follows –

$$S = \begin{pmatrix} g^{(\vec{1})} - d^{(\vec{1})} \\ g^{(\vec{2})} - d^{(\vec{2})} \\ \vdots \\ g^{(\vec{n})} - d^{(\vec{n})} \end{pmatrix} \tag{2.16}$$

Now once we have the matrix S, then we have –

$$\theta(d|m) = \frac{1}{(2\pi^n|C_\Delta|)^{1/2}} exp\left(\frac{-S^T C_\Delta^{-1} S}{2}\right). \tag{2.17}$$

where the C_Δ is the co-variance matrix.
Now once the $\theta(d|m)$ is constructed we have the likelihood function given as,

$$L(m) = \theta(d|m)\rho_M(m), \tag{2.18}$$

where the $\rho_M(m)$ is the prior information over the model space, generally some guassian distribution for each parameter in the model space. Now we are ready to run a MCMC simulation for estimating the parameters in the model space.

Chapter 3

Conclusions

Many physical and biological processes can be represented using a mathematical model by abstracting a large number of processes into mathematical functions. These models are useful in understanding the behaviour of the system under different conditions, which are generally performed by changing the parameters. When these real systems are subjected to experimentation, one gets data on the observables (variables) under different conditions. It is an important task to extract the parameters from the data set that may be responsible for the behaviour observed in experiments. Here we have studied a well established and complex mathematical and statistical approach for determining the model space parameters from given data. The approach involves Inverse Problem Theory and application of the Monte Carlo Markov Chain (MCMC) techniques. We have described the methods involved and then applied them to two examples of increasing complexity. First we study the simple problem of estimating two parameter of an Ellipse from a given dataset and some prior information about the possible values of the parameters. Our results show convergence to good estimated parameter values on MCMC analysis. Then we applied the problem of estimation to the complex system of the Lorenz equations, which shows a variety of complex dynamics - from equilibrium to chaotic - for different parameter values. We have given the theoretical methodology involved in the estimation and further work is in progress to extract reasonable parameter values from the data. We have also attempted to apply the method on a highly nonlinear coupled discrete system of equations describing the population dynamics of insects, which involves 7 parameters. Work is in progress in that area too.

Bibliography

[AS09] S R Athreya and VS Sunder, *Measure and Probability*, Universities Press (India) Private Limited, 2009.

[BN06] Krishna B.Athreya and Soumendra N.Lahari, *Probability Theory*, Hindustan Book Agency (India), 2006.

[Tar05] Albert Tarantola., *Inverse Problem Theory*, Society for Industrial and Applied Mathematics, Phildelphia, 2005.

Appendix A

Ellipse Problem

```
/*
Information for about the plots created...
There are four plots that are obtained
after running the code...
1)The likelihood_plot.txt for the likelihood
  function... z=likelihood(a,b)
2)The time step process for parameter a...
  which is a_t.txt........mcmc plot for a...
3)The time step process for parameter b...
  which is b_t.txt........mcmc plot for b...
4)The plot for r,a and b.....
  which is called stationary_distribustion_plot.txt...
  which is not really the stationary
  distribution but just to make observations...
  which in this case looks like kind of gaussian... !!
*/



#include<stdio.h>
#include<math.h>
#include"jvrand.h"

#define rounds 100000
// number of rounds (or) the number of times the a and b are fetched randomly...
#define beta 0.3
// beta is a constant used in fun_Q to avoid the symmetry in transition probablility...
#define robs 1
// robs is the standard deviation observed in the measurement of x co-ordinate value
#define delta_Q 1
// standard deviation for the fun_Q densit function.....
#define pi 3.14159
// pi constant.... as usual ...
#define mean_a 1.81463
// mean_a is the mean value for 'a' used in the nu_M .... M is the model space...
// note : nu() function is not in the program.. it is the prior probability density
//function just wrote down in the return value of the lklhood_fun() which is defined in the program...
#define mean_b 0.503165
// mean_b is the mean value for 'b' used in the nu_M..
#define delta_a pow(0.5,2)
// delta_a is standard deviation for 'a' ... actually the variance...
#define delta_b pow(0.2,2)
// delta_b is standrad deviation for 'b '.... actually the variance...
#define beta_a 2
// test values for fun_Q....
#define beta_b 2
// test values for fun_Q....
double data[100];
// for the co-ordinates in the inputFile... all even data[]'s are x co-ordinates and odd are y
```

```
// co-ordinates..... change the array parameter according to the input ....


double fun_Q(double next_a , double a, double next_b, double b)
// the density function from which the 'a' and 'b' samples are picked randomly ...
//(a,b) is the present state... (next_a,next_b) is the next state to be choosen or not...
{
   return  exp(-pow((next_a-(beta*a)),2)/2*pow(delta_Q,2))*exp(-pow((next_b-(beta*b)),2)/2*pow(delta_Q,2));
}



double lklhood_fun(double A, double B )
// the likelihood function which is directly proportional to the product of the priori
//information(nu_M function) and posterior probability (the theta function)..
{
double sum ;
sum = -0.5*pow((abs(data[0])-abs(A*(sqrt(abs(1-(pow(data[1],2)/pow(B,2)))))),2)/pow(robs,2);
// data[] is globally declared ....
// the abs(x)-abs(.......) is used , so no need to worry about the sign of  the abs(........).
int j;
for (j=2;j<=98;j+=2)
{
   sum =sum + (-0.5*pow((abs(data[j])-abs(A*(sqrt(abs(1-(pow(data[j+1],2)/pow(B,2)))))),2)/pow(robs,2)) ;
                        }
return exp(sum)*exp((-(pow(A-mean_a,2))/2*delta_a)+(-pow(B-mean_b,2)/2*delta_b));
// exp(sum) is the posterior probability
// and the rest is the prior probability density....

}

main()
{
if (remove("likelihood_plot.txt")!=0 && remove("a_t.txt")!=0 &&
    remove("b_t.txt")!=0 && remove("stationary_distribution_plot.txt")!=0)
perror("the_old_output_files_could_not_be_deleted_or_does_not_exist\n");
else
printf("the_old_output_files_are_deleted\n");

FILE *fp1;
double c;
fp1 = fopen("ellipse_data3.txt","r");
do {
c = fscanf(fp1,"%lf", data);
} while (c != EOF);
fclose(fp1);


double array_a[rounds],array_b[rounds];
// storing the values of 'a' and 'b' for further use .... actually they are the values
// for next_a and next_b subjected to test....
double a,b;
// these represent the present state of the model space at any point of the program ...
int accepted_a_b=0;
// to keep count of number of states accepted.........

printf("enter_the_initial_value_for_a_:");
scanf("%lf",&a);
printf("\n");
printf("enter_the_initial_value_for_b_:");
scanf("%lf",&b);
printf("\n");

FILE *fp2,*fp_a,*fp_b,*fp_3d;

int i;
double z ;
// variable for the likelihood function .....
for (i=0;i<=rounds;i++)
{
array_a[i] = jvrand_normal (beta*a,delta_Q);
// generating next_a from fun_Q where 'beta*a' is mean...
array_b[i] = jvrand_normal (beta*b,delta_Q);
```

```
// generating next_b  ..................................
z=lklhood_fun(array_a[i],array_b[i]);
// calculating the value of likelihood function at generated a and b values ......

fp_3d =fopen("likelihood_plot","a");
fprintf(fp_3d,"%lf\t%lf\t%lf\n",array_a[i],array_b[i],z);
fclose(fp_3d);

double r=(lklhood_fun(array_a[i],array_b[i])*fun_Q(array_a[i],a,array_b[i],b))
                        /(lklhood_fun(a,b)*fun_Q(a,array_a[i],b,array_b[i]));
// the ratio test ... and it is stored as 'r '......

double u;
u =jvrand_uniform(0.0,1.0);
// generating random number between 0 and 1 ......

printf("%d\t%lf\t%lf\n",i,a,b);
 if (r>=u)
// the test to accept next_a and next_b or not ........
// if true .. update them as the new present states
// for 'a' and 'b' .........
  {    a=array_a[i];
       b=array_b[i];
accepted_a_b++;
fp2= fopen("stationary_distribution_plot.txt","a");
fprintf(fp2,"%lf\t%lf\t%lf\n",a,b,r);
fclose(fp2);                }



fp_a= fopen("a_t.txt","a");
fprintf(fp_a,"%d\t%lf\n",i,a);
fclose(fp_a);

fp_b= fopen("b_t.txt","a");
fprintf(fp_b,"%d\t%lf\n",i,b);
fclose(fp_b);
}

int p;
double sum_a= array_a[0];  // for the mean of 'a' ...
for (p=1;p<=rounds;p++)
{
  sum_a=sum_a+array_a[p];     }

double sum_b= array_b[0];      // for the mean of 'b' ....
for (p=1;p<=rounds;p++)
{
  sum_b=sum_b+array_b[p];     }

printf("%lf\t%lf\t%d\t%d\n",a,b,accepted_a_b,rounds);
printf("%lf\t%lf\n",sum_a/rounds,sum_b/rounds);

return 0;
}
```

	x	y		x	y
1	1.5938954e+00	2.2174030e-01	26	-1.1943160e+00	3.5405783e-01
2	6.1403762e-01	4.5206091e-01	27	-1.1750244e+00	3.7408274e-01
3	1.2195540e+00	3.7615132e-01	28	-1.4051071e+00	3.0978135e-01
4	1.1826470e+00	3.8581423e-01	29	-1.3233742e+00	3.2793660e-01
5	4.4391760e-01	4.8555695e-01	30	-7.7593299e-01	4.4837960e-01
6	3.7849652e-01	4.8337576e-01	31	-9.9006673e-01	4.1055471e-01
7	9.4754608e-01	4.2207883e-01	32	-1.1725107e+00	3.5885129e-01
8	1.6396159e+00	2.1231935e-01	33	-4.5375706e-01	4.8739075e-01
9	7.1057859e-01	4.6429053e-01	34	-8.2210386e-01	4.5303999e-01
10	1.0779016e+00	4.0783013e-01	35	-8.7541459e-01	4.5265500e-01
11	5.3571791e-01	4.9446092e-01	36	-3.2420950e-01	4.8674291e-01
12	1.3269006e+00	3.3979654e-01	37	-1.2712415e+00	3.5680347e-01
13	5.8264267e-01	4.5169798e-01	38	-5.6419966e-01	4.7513810e-01
14	9.5893558e-01	4.1474297e-01	39	-5.6940636e-01	4.8766024e-01
15	1.2486151e+00	3.4659810e-01	40	-1.1293312e+00	4.0062098e-01
16	1.5363549e+00	2.4979990e-01	41	-8.4826754e-01	4.3749544e-01
17	1.6389371e+00	1.9712228e-01	42	-1.5862186e+00	2.3933216e-01
18	1.0208233e+00	4.1057570e-01	43	-1.6190748e+00	2.1824588e-01
19	4.0793666e-01	5.0135726e-01	44	-9.0380367e-01	4.2978101e-01
20	4.2394101e-01	4.6632541e-01	45	-5.3124915e-01	4.6022513e-01
21	5.8626238e-01	4.7075937e-01	46	-2.9898397e-01	4.9591078e-01
22	1.4610759e+00	2.7995270e-01	47	-1.5051407e+00	2.8253415e-01
23	5.8142327e-01	4.4411720e-01	48	-8.4676451e-01	4.5101213e-01
24	1.4214272e+00	3.1501050e-01	49	-9.9591404e-01	4.0493180e-01
25	5.6528745e-01	4.8849318e-01	50	-1.6821469e+00	1.7261179e-01

Table A.1: Data- Ellipse XY co-ordinates