Bayesian calibration of stochastic kinetic models using a Dirichlet process mixture of Gaussian processes

Aamir Khan

Thesis submitted for the degree of
Doctor of Philosophy

School of Mathematics, Statistics & Physics
Newcastle University
Newcastle upon Tyne
United Kingdom

March 2020

Abstract

Stochastic kinetic models (SKMs) are an effective way to model complex biochemical and cellular systems. They describe how a number of species in a system interact with one another through time. To infer the parameters of these models, a number of MCMC techniques exist but these can often be both computationally intensive and time consuming due to the constant need to simulate from the stochastic process at each iteration. When inferring parameters of quite large or complex models, these simulations can become unmanageable.

To tackle this, emulators can be used to approximate SKM output, a popular choice being a Gaussian process, however these do not provide accurate descriptions of output with multiple modes. A SKM of particular interest which exhibits this behaviour is the Schlögl system which describes an exchange of chemicals between two material baths. This system, under certain conditions, is bistable.

This motivates the need to find a flexible emulator that can capture this bimodality. By using a Dirichlet process mixture of Gaussian processes we explain how this model has useful features such as the flexibility to increase or decrease the number of components in the mixture throughout parameter space as necessary.

We apply the model to training data for the Schlögl system with the aim of inferring the rate constants that gave rise to some noisy data from the system. We also look at a further approximation using variational inference and find that this gives significant gains in terms of efficiency.

Acknowledgements

I am immensely grateful to my supervisors Richard Boys and Andrew Golightly for their advice, guidance and support without which I would not have been able to complete my studies.

Richard has helped shape my career and he was a mentor ever since my time as an undergraduate at Newcastle University. I would like to thank him for being there for me as a supervisor and friend who I will miss greatly.

I am immeasurably thankful to Andy for his time and support during the rather difficult recent months and for helping me finish this thesis.

Thank you to my friends and colleagues in the School of Mathematics, Statistics & Physics for your support throughout my studies. Special thanks to Stephen, Joe and David for the in-depth discussions and debugging sessions.

I would also like to express my gratitude to the Engineering and Physical Sciences Research Council for the financial support which made the writing of this thesis possible.

Thank you to my friends Andy, John and Mike who have given me laughs over the years and acted interested when I've told them all about my thesis. Naseem and Elliott — thank you for your daily support, laughter and encouragement. Michelle and Anthony — thank you for your love and support (and many takeaways) and frequently reminding me how bad I am at quick mental arithmetic. And to the rest of my family: Shaheen, Samena, Phil, Jan and my grandmother — thank you for your love, laughter and support and also for your enthusiasm whenever I've explained what I've been doing all these years. Special thanks to my mum for her love and kindness and for asking regularly if I've "finished that PhD yet".

Last but certainly not least, I would like to thank my wife Amy for her endless love, patience and support every single day — you have given me strength and motivation and you have helped me more than you will ever know.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

A stochastic kinetic model (SKM) typically refers to a reaction network and a stochastic description of the dynamics of the species in the network. The most natural Markov jump process (MJP) representation has been ubiquitously applied to describe epidemics (Lin and Ludkovski (2014); McKinley et al. (2014)), predator-prey interactions (Boys et al., 2008) and in systems biology (Golightly and Wilkinson (2011); Owen et al. (2015); Golightly and Sherlock (n.d.)). Of particular interest in this thesis is the Schlögl system (Schlögl, 1972), where three different species interact between two material baths. This system exhibits nonlinear dynamics, and, for particular parameter choices, has two stable states.

Whilst forward simulation of MJPs is straightforward, using for example Gillespie's direct method (Gillespie, 1976), the reverse problem of inferring the parameters governing the MJP can be a challenging problem. In practice, we expect data at discrete times that may be incomplete (in the sense that only a subset of species counts are observed) and subject to measurement error. Calculating the observed data likelihood requires summing over a potentially infinite number of states rendering this calculation practically intractable. Bayesian inference can in principle take place via computationally intensive methods such as Gibbs sampling (Boys et al., 2008) or particle Markov chain Monte Carlo (Andrieu et al. (2010) and Golightly and Wilkinson (2011)). Such methods typically require the equivalent of many millions of runs of the forward simulator rendering them computationally infeasible for SKMs with many reactions and species. Consequently, several authors have investigated the use of surrogate models whereby exact (simulation-based) inference is performed using the surrogate. We briefly review the most well studied approaches.

The chemical Langevin equation or diffusion approximation (Gillespie, 2000) can be found by matching the first two infinitesimal moments of the MJP to the drift and diffusion functions of an Itô stochastic differential equation (SDE). Hence, discreteness (but not stochasticity) is ignored. The resulting SDE is typically intractable but can be discretised,

with the choice of discretisation controlling computational cost (Golightly and Wilkinson (2005); Golightly and Wilkinson (2006)). Nevertheless, the observed data likelihood remains intractable, again necessitating the use of (particle) MCMC (Andrieu et al., 2010). The SDE can be further approximated by linearising the drift and diffusion functions to give the linear noise approximation (LNA, Van Kampen (1992); Komorowski et al. (2009)). The resulting process has Gaussian transition densities (for fixed or Gaussian initial conditions) and therefore admits a tractable observed data likelihood (under the assumption of additive Gaussian noise, as is often used in practice). Although the LNA is computationally inexpensive, it is well known to give a poor approximation in low count number scenarios (Schnoerr et al., 2017). Moreover, it is unable to accommodate SKMs that give multi-modal output.

Another approach is to regard the output of the SKM as that of an expensive computer model, allowing the applications of methods from the *emulation* literature. See e.g. Sacks et al. (1989), Currin et al. (1991) and Kennedy and O'Hagan (2001). The essential idea is to generate *training data* by repeatedly running the SKM simulator (e.g. Gillespie's direct method) for various choices of parameter values (the inputs, although time may also be an input) to obtain realisations of the SKM at each observation time (the outputs). As shown in Henderson et al. (2009) and Henderson et al. (2010) a Gaussian process (GP) is used to model the mean and variance, which are smooth nonlinear deterministic functions of the inputs. This then allows for straightforward prediction of the mean and variance for parameter values that do not appear in the training set. We may anticipate that a GP is unable to adequately account for heavy tailed and/or multi-model output exhibited by some SKMs e.g. the Schlögl system which is bimodal as a function of the parameters.

A typical solution to modelling multi-modal output is to fit a mixture model, which may also allow for heavier tails. A simple approach to mixture modelling is to pre-specify the number of components *a priori*. For example, we might assume that two components would be adequate for the Schlögl system. However, given a complex reaction network and a data poor scenario, pre-specifying a satisfactory number of mixture components may be difficult. We can alleviate this issue by using a Dirichlet process mixture (MacEachern and Müller (1998) and Neal (2000)), where the number of mixture components can be inferred from the data.

The main contribution of this thesis is the development and fitting of an emulator that is a Dirichlet process mixture of Gaussian processes. We believe that this is the first use of this approach within the SKM setting. In addition, we compare and contrast this approach with standard likelihood-free approaches from the literature. We illustrate the proposed methodology using the Schlögl system, for which exact simulation based inference under the MJP is computationally demanding. The remainder of the thesis is organised as

follows.

1.1 Outline of thesis

We begin in Chapter 2 by describing the principles of stochastic modelling and defining notation used for chemical reactions. We describe the Markov jump process representation of SKMs as well as explain how to simulate from these models via Gillespie's direct method. We give examples of SKMs, starting with a simple birth-death model and concluding with the more complex Schlögl system.

We will adopt a Bayesian approach to make inferences from the data and find the posterior distribution of the parameters of the SKM. Thus in Chapter 3 we give an overview of the Bayesian approach to inference and state Bayes' theorem. For most problems of practical interest, the parameter posterior is likely to be intractable, necessitating the use of computationally intensive methods such as MCMC. We then move on to explain how to perform inference when given noisy data assumed to come from a SKM at a number of discrete time points. The observed-data likelihood will be intractable for all but the simplest SKMs, and we therefore consider likelihood-free methods such as likelihood-free MCMC and sequential Monte Carlo (SMC), including particle filters.

Gaussian processes (GPs) are then defined in Chapter 4 as we begin to explore how to construct the flexible emulator of SKM output. We give an overview of how a GP can be used for regression and define the mean function and covariance function. We conclude this chapter with a simulation study that concerns inferring the hyperparameters of the GP covariance function. In Chapter 5 we explore the use of GPs for emulation. A key aspect of GP emulation is choice of appropriate input data. We explore the use of a Latin hypercube design as a method for giving efficient coverage of the input space. We look at an example where a GP is used to emulate SKM output as well as diagnostics.

Since we wish to build a mixture of GPs to emulate SKM output we introduce finite mixture modelling in Chapter 6. The Dirichlet process (DP) is then considered. This is defined formally and, additionally, through a number of representations to help with understanding including the stick-breaking process, Chinese restaurant process and Pólya urn scheme. The use of DPs in mixture modelling is then discussed and we explain how this can be done using MCMC methods. Two examples are given at the end of this chapter where we fit a DP mixture (DPM) of normals to data simulated from a three-component normal mixture to help demonstrate how adaptable a DPM can be. The second example builds upon this by adding in spatial dependence as we consider fitting a DPM of regression lines.

Chapter 7 defines a Dirichlet process mixture of Gaussian processes (DPMGP) model. We explain how to efficiently evaluate the log-likelihood function for this model by using efficient tools to invert the large correlation matrix and find the determinant. We show how each point in the data is assigned to a cluster and explain how we can perform prediction for this model. We conclude the chapter by fitting the model to data obtained from a two-component mixture of GPs.

In Chapter 8 we focus on finding the posterior distributions of the stochastic rate constants for the Schlögl system given noisy data at a small number of time points. We consider inference using Gillespie's direct method as well as using our proposed DPMGP emulator approach. In Chapter 9, we consider a fast but approximate inference scheme for the DPMGP. Essentially, we consider the variational approach of Ross and Dy (2013) and adapt it to our setting. Conclusions and possible future research avenues are given in Chapter 10.

Chapter 2

Stochastic kinetic models

2.1 Introduction

This chapter introduces the concept of stochastic kinetic models. We introduce notation used for chemical reactions which describe stochastic kinetic models and which are a natural way to model complex biochemical and cellular systems.

We describe a method to simulate from stochastic kinetic models, specifically defining Gillespie's direct method. We introduce three different stochastic kinetic models: the simple birth-death model, the Lotka-Volterra model and the more complex Schlögl system.

For further detail of concepts discussed in this chapter see Golightly and Gillespie (2013), Schnoerr et al. (2017) and Wilkinson (2018).

2.2 Chemical reactions

In order to define stochastic kinetic models and the associated theory, we first consider a biochemical system in which there are n species denoted by $X_1, \ldots, X_n$. How each of the molecules interact with each other through time is described by a collection of chemical reactions where we denote the set of m reactions by $R_1, \ldots, R_m$.

When a reaction occurs, the level (count) of one or more of the species changes, for example, an X_1 molecule and an X_2 molecule might react with each other to produce an X_3 molecule. This would mean the counts of molecules for species X_1 and X_2 would both decrease by one and the count of molecules for species X_3 would increase by one. We denote this chemical reaction using the following equation

$$X_1 + X_2 \xrightarrow{c} X_3.$$

The species on the left, X_1 and X_2, are called the reactants and the specie X_3 is the product. This reaction, under certain assumptions (see Section 2.3), will occur according to some rate constant c. We now generalise this to define a network of reactions.

A network of reactions is given by a collection of m reaction equations, which describe how the n species of the system react with one another over time, where a typical reaction is written

$$\begin{aligned}
R_1: &\quad a_{11}X_1 + \ldots + a_{1n}X_n &&\xrightarrow{c_1}\quad b_{11}X_1 + \ldots + b_{1n}X_n,\\
R_2: &\quad a_{21}X_1 + \ldots + a_{2n}X_n &&\xrightarrow{c_2}\quad b_{21}X_1 + \ldots + b_{2n}X_n,\\
&\quad\vdots &&\quad\vdots\\
R_m: &\quad a_{m1}X_1 + \ldots + a_{mn}X_n &&\xrightarrow{c_m}\quad b_{m1}X_1 + \ldots + b_{mn}X_n,
\end{aligned}$$

where the c_i, $i = 1, \ldots, m$ are the kinetic rate parameters of the system. The vector of rate constants is denoted by $\boldsymbol{c} = (c_1, \ldots, c_m)^\top$. Note that it is often more convenient to work with the log stochastic rates, $\boldsymbol{\theta} = \log \boldsymbol{c}$.

The $\{a_{ij}\}$ and $\{b_{ij}\}$ are known as stoichiometric coefficients, where a_{ij} gives the number of reactants of type j in reaction i, with b_{ij} giving the number of products. If we write $A = (a_{ij})$ and $B = (b_{ij})$ as the $m \times n$ matrices of stoichiometric coefficients then $Q = B - A$ gives the net effect matrix. That is where the ij^{th} element describes how a type R_i reaction changes the count of type X_j species. We often work with the transpose of this net effect matrix, and define the stoichiometry matrix S by

$$S = Q^\top = (B - A)^\top.$$

As with the net effect matrix, the stoichiometry matrix encodes other important information about the reaction network. For example, vectors in the left null-space of S correspond to conservation laws in the network, allowing the study of the dynamics of a reduced number of species which we will see in Section 2.7.

At time t, we denote the number of molecules of type X_j in the system by $x_j(t)$, where the state of the entire system is denoted by $\boldsymbol{x}(t) = (x_1(t), \ldots, x_n(t))^\top$.

2.3 Markov jump process

Some assumptions are made about the molecules, mainly that they are in a container of fixed volume which is well stirred and in thermal equilibrium. Under these assumptions the movement of the molecules is random and driven by Brownian motion.

It can then be shown (Gillespie, 1992) that the rate of reactions is constant over a small interval of time δt and as such each reaction will have an associated stochastic rate constant which we previously denoted by c_i for reaction R_i. Recall that we work with the log stochastic rates, $\boldsymbol{\theta} = \log \boldsymbol{c}$. We assume mass-action kinetics, which states that the rate of the reaction is proportional to the product of the concentrations of the reactants. Under this assumption, the rate constant together with the state of the system at time t, $\boldsymbol{x}(t)$, defines the hazard function $h_i(\boldsymbol{x}(t), \boldsymbol{\theta})$ of reaction R_i. Thus, the instantaneous rate of reaction R_i at time t is $h_i(\boldsymbol{x}(t), \boldsymbol{\theta})\delta t$ in a small time interval δt.

For example, a first order reaction would be of the form

$$X \xrightarrow{e^\theta} 2X.$$

The instantaneous hazard of this reaction happening in an interval δt is $e^\theta \delta t$. If there are x molecules of species X at time t then the hazard of this reaction, under mass-action stochastic kinetics, would be

$$h(x, \theta) = e^\theta x.$$

Consider a second order reaction of the form

$$X_1 + X_2 \xrightarrow{e^\theta} X_3.$$

This would occur when a molecule of type X_1 collides with a molecule of type X_2. If there are x_1 molecules of species X_1 and x_2 molecules of species X_2 then the hazard of this reaction is

$$h(\boldsymbol{x}, \theta) = e^\theta x_1 x_2.$$

The reaction

$$2X_1 \xrightarrow{e^\theta} X_2$$

is also second order and occurs when two molecules of type X_1 collide with each other. The hazard of this reaction is

$$h(\boldsymbol{x}, \theta) = e^\theta \frac{x_1(x_1 - 1)}{2!}.$$

By considering the number of ways in which the reactants (on the left hand side of the reaction) can react with each other we can generalise the hazard function to be proportional to a product of binomial coefficients with

$$h_i(\boldsymbol{x}(t), \theta_i) = e^{\theta_i} \prod_{j=1}^{n} \binom{x_j(t)}{a_{ij}}. \tag{2.1}$$

The total rate (or hazard) of any reaction occuring is simply the sum of hazards,

$$h_0(\boldsymbol{x}(t), \boldsymbol{\theta}) = \sum_{i=1}^{m} h_i(\boldsymbol{x}(t), \theta_i) = \sum_{i=1}^{m} e^{\theta_i} \prod_{j=1}^{n} \binom{x_j(t)}{a_{ij}}.$$

The effect of reaction i results in a change to the system state $\boldsymbol{x}(t)$ abruptly and discretely. If reaction i were to occur at time t, the new state is

$$\boldsymbol{x}(t) = \boldsymbol{x}(t_-) + S^i$$

where S^i is the i^{th} column of the stoichiometry matrix S and $\boldsymbol{x}(t_-)$ is the state of the system at the previous time. The time evolution of $\boldsymbol{x}(t)$ is then described by a continuous-time, discrete-valued Markov process.

Assuming an initial state $\boldsymbol{x}_0$, the state of the system at time t is

$$\boldsymbol{x}(t) = \boldsymbol{x}_0 + \sum_i S^i R_{i,t}$$

where $R_{i,t}$ denotes the number of times the i^{th} reaction has occurred by time t. The process $R_{i,t}$ is a counting process with intensity $h_i(\boldsymbol{x}(t), \theta_i)$, the hazard function defined above in (2.1). The counting process can be written as

$$R_{i,t} = Y_i \left(\int_0^t h_i(\boldsymbol{x}(s), \theta_i) \, \mathrm{d}s \right)$$

where $Y_i, i = 1, \ldots, m$ are independent unit rate Poisson processes. We refer the reader to Kurtz (1972) and Wilkinson (2018) for more details on this representation.

2.4 Simulation methods and approximations

A number of simulation methods exist to generate realisations from stochastic kinetic models. They can be computationally expensive depending on the particular system involved. For example, systems with high levels of species will typically have more reactions occurring as a result of many species interacting with each other. Consequently a number of approximation methods exist which aim to take less time than exact methods, at the expense of some accuracy.

Algorithm 1 Gillespie's direct method

1. At time $t = 0$, set the initial state to be $\boldsymbol{x}(0)$.
2. Calculate the hazard $h_i(\boldsymbol{x}(t), \theta_i)$ for each reaction $i = 1, \ldots, m$ and the combined hazard $h_0(\boldsymbol{x}(t), \boldsymbol{\theta}) = \sum_{i=1}^{m} h_i(\boldsymbol{x}(t), \theta_i)$.
3. Simulate the dwell time $\delta_t \sim Exp(h_0(\boldsymbol{x}(t), \boldsymbol{\theta}))$ and set $t := t + \delta_t$.
4. Simulate the index of the next occurring reaction j from a discrete distribution with probabilities $h_i(\boldsymbol{x}(t), \theta_i)/h_0(\boldsymbol{x}(t), \boldsymbol{\theta})$ for $i = 1, \ldots, m$.
5. Update the state of the system $\boldsymbol{x}$ according to reaction j.
6. If $t < T$ return to step 2.

2.4.1 Gillespie's direct method

The most common simulation method is the Gillespie algorithm (Gillespie, 1976) which is also known as the direct method. This is an exact stochastic simulation method that allows us to generate exact trajectories from the MJP representation of a stochastic kinetic model. It is an iterative scheme where, after initialisation, the dwell time (the time that the process remains in the current state) is an exponential random variable with rate given by the sum of the hazards of each reaction given the current specie levels, which we earlier denoted by $h_0(\boldsymbol{x}(t), \boldsymbol{\theta})$, the total rate. The reaction which next occurs is then simulated from a discrete distribution with probabilities being the ratio of the hazard of reaction i and $h_0(\boldsymbol{x}(t), \boldsymbol{\theta})$. That is

$$\frac{h_i(\boldsymbol{x}(t), \theta_i)}{h_0(\boldsymbol{x}(t), \boldsymbol{\theta})} \quad \text{for } i = 1, \ldots, m.$$

The algorithm reaches completion when some time T is reached. The full details of the algorithm are given in Algorithm 1.

2.4.2 Chemical Langevin equation

We now present an overview of the chemical Langevin equation (CLE) and refer the reader to Gillespie (2000) for further details.

We begin by considering an infinitesimal time interval, $(t, t + dt]$, over which the reaction hazards will remain constant almost surely. The occurrence of reaction events can then be regarded as the occurrence of events of a Poisson process with independent realisations for each reaction type. If, over the infinitesimal time increment, we have the number of reaction events of each type in the m-vector $d\boldsymbol{R}(t)$, it is clear that the elements

are independent of one another and that the i^{th} element is a $Po(h_i(\boldsymbol{x}(t), \theta_i)dt)$ random variable. Thus, we have the expectation and variance as $E(d\boldsymbol{R}(t)) = h(\boldsymbol{x}(t), \boldsymbol{\theta})dt$ and $Var(d\boldsymbol{R}(t)) = \text{diag}\{h(\boldsymbol{x}(t), \boldsymbol{\theta})\}dt$. It follows that the Itô stochastic differential equation (SDE)

$$d\boldsymbol{R}(t) = h(\boldsymbol{x}(t), \boldsymbol{\theta})dt + \text{diag}\left\{\sqrt{h(\boldsymbol{x}(t), \boldsymbol{\theta})}\right\} d\boldsymbol{W}(t)$$

has the same infinitesimal mean and variance as the true Markov jump process (where $d\boldsymbol{W}(t)$ is the increment of a m-dimensional Brownian motion). Now since $d\boldsymbol{X}(t) = Sd\boldsymbol{R}(t)$, we obtain

$$d\boldsymbol{X}(t) = S\, h(\boldsymbol{x}(t), \boldsymbol{\theta})dt + \sqrt{S\text{diag}\{h(\boldsymbol{x}(t), \boldsymbol{\theta})\}S'}d\boldsymbol{W}(t), \tag{2.2}$$

where now $\boldsymbol{X}(t)$ and $\boldsymbol{W}(t)$ are both n-vectors. Equation (2.2) is the SDE commonly referred to as the chemical Langevin equation and represents the diffusion process which most closely matches the dynamics of the associated Markov jump process. In high concentration scenarios the CLE approximates the stochastic kinetic models well (Gillespie, 2000). In the absence of an analytic solution to (2.2), a numerical solution can be constructed. For example, the Euler-Maruyama approximation is

$$\begin{aligned}\Delta\boldsymbol{X}(t) &\equiv \boldsymbol{X}(t + \Delta t) - \boldsymbol{X}(t) \\ &= S\, h(\boldsymbol{x}(t), \boldsymbol{\theta})\Delta t + \sqrt{S\text{diag}\{h(\boldsymbol{x}(t), \boldsymbol{\theta})\}S'}\Delta\boldsymbol{W}(t)\end{aligned} \tag{2.3}$$

where Δt is a small interval of time and $\Delta\boldsymbol{W}(t)$ is a mean zero Normal random vector with variance matrix $\text{diag}\{\Delta t\}$. Higher order approximations are possible; see e.g. Kloeden and Platen (2013).

Performing exact (simulation based) inference for the diffusion approximation has been the focus of many authors including Golightly and Wilkinson (2005), Purutcuoglu and Wit (2007), and Golightly and Wilkinson (2011). Although Golightly and Wilkinson (2011) find that a particle MCMC scheme based on the CLE can be more computationally efficient than a similar scheme that works with the Markov jump process directly, the particle MCMC scheme requires calculation of an estimate of marginal likelihood under the CLE which can be computationally expensive, depending on the required discretisation.

2.4.3 Linear noise approximation

The LNA was first considered as a functional central limit law for density dependent processes by Kurtz (1970) and can be derived in a number of ways. For example, Komorowski et al. (2009) and Elf and Ehrenberg (2003) derive the LNA by approximating the forward Kolmogorov equation (satisfied by the transition rate of the MJP) through a Taylor series

expansion. The first and second moments are used in the approximation however further moments can be used (Ale et al., 2013). We shall give a more informal derivation similar to that of Fearnhead et al. (2014) and we refer the reader to the references therein for a more detailed discussion. In what follows we calculate the LNA for a general SDE before formulating it as an approximation to the CLE.

Consider a general SDE satisfied by a process $\{\boldsymbol{X}(t), t \geq 0\}$ of the form

$$d\boldsymbol{X}(t) = \alpha(\boldsymbol{X}(t))dt + \epsilon\beta(\boldsymbol{X}(t))d\boldsymbol{W}(t) \tag{2.4}$$

where $\epsilon << 1$. Partition $\boldsymbol{X}(t)$ into a deterministic path $\boldsymbol{z}(t)$ and a residual stochastic process $\boldsymbol{M}(t)$ and let $\boldsymbol{z}(t)$ be the solution to

$$\frac{d\boldsymbol{z}(t)}{dt} = \alpha(\boldsymbol{z}(t)). \tag{2.5}$$

We assume that $||\boldsymbol{X}(t) - \boldsymbol{z}(t)||$ is $O(\epsilon)$ over a time interval of interest, where $||\cdot||$ is the Euclidean norm, and substitute $\boldsymbol{X}(t) = \boldsymbol{z}(t) + \epsilon\boldsymbol{M}(t)$ into equation (2.4) to give

$$d(\boldsymbol{z}(t) + \epsilon\boldsymbol{M}(t)) = \alpha(\boldsymbol{z}(t) + \epsilon\boldsymbol{M}(t))dt + \epsilon\beta(\boldsymbol{z}(t) + \epsilon\boldsymbol{M}(t))d\boldsymbol{W}(t).$$

We then Taylor expand $\alpha(\cdot)$ and $\beta(\cdot)$ about $\boldsymbol{z}(t)$ and collect terms of $O(\epsilon)$ to give the SDE satisfied by $\boldsymbol{M}(t)$ as

$$d\boldsymbol{M}(t) = F(t)\boldsymbol{M}(t)dt + \beta(\boldsymbol{z}(t))d\boldsymbol{W}(t) \tag{2.6}$$

where $F(t)$ is the Jacobian matrix with (i,j)th element $\partial\alpha_i(\boldsymbol{z}(t))/\partial z_j(t)$ and $\alpha_i(\boldsymbol{z}(t))$ refers to the ith element of $\alpha(\boldsymbol{z}(t))$. Note the use of ϵ to indicate that the stochastic term in (2.4) is small and, essentially, that the drift term dominates the diffusion coefficient. However ϵ plays no role in the evolution equations, (2.5) and (2.6). Without loss of generality, therefore, we simplify by setting $\epsilon = 1$. To further simplify the notation we also drop the explicit dependence of the hazard function on $\boldsymbol{\theta}$, and of the mean and variance of $\boldsymbol{M}(t)$ on both $\boldsymbol{\theta}$ and $\boldsymbol{z}(t)$.

For the CLE, we have

$$\alpha(\boldsymbol{X}(t)) = S\,h(\boldsymbol{X}(t)), \qquad \beta(\boldsymbol{X}(t)) = \sqrt{S\text{diag}\{h(\boldsymbol{x}(t), \boldsymbol{\theta})\}S'}.$$

The linear noise approximation of the CLE is therefore defined through

$$\frac{d\boldsymbol{z}(t)}{dt} = Sh(\boldsymbol{z}(t), \boldsymbol{\theta}) \tag{2.7}$$

and

$$d\boldsymbol{M}(t) = F(t)\boldsymbol{M}(t)dt + \sqrt{S\text{diag}\{h(\boldsymbol{z}(t), \boldsymbol{\theta})\}S'}d\boldsymbol{W}(t) \tag{2.8}$$

where $F(t)$ has (i,j)th element $S\partial h_i(\boldsymbol{z}(t),\theta_i)/\partial z_j(t)$.

For fixed or Gaussian initial conditions, that is $\boldsymbol{M}(t_1) \sim N(\boldsymbol{m}(t_1), V(t_1))$, the SDE in (2.8) can be solved explicitly to give

$$(\boldsymbol{M}(t)|\boldsymbol{\theta}) \sim N\left(\boldsymbol{m}(t)\,,\, V(t)\right) \tag{2.9}$$

where $\boldsymbol{m}(t)$ is the solution to the deterministic ordinary differential equation (ODE)

$$\frac{d\boldsymbol{m}(t)}{dt} = F(t)\boldsymbol{m}(t) \tag{2.10}$$

and similarly

$$\frac{dV(t)}{dt} = V(t)F(t)' + S\text{diag}\{h(\boldsymbol{z}(t),\boldsymbol{\theta})\}S' + F(t)V(t)\,. \tag{2.11}$$

Hence, the solution of equation (2.8) requires the solution of a system of coupled ODEs given by (2.7), (2.10) and (2.11). In the absence of an analytic solution to these equations, a numerical solution can be used. The approximating distribution of $\boldsymbol{X}(t)$ can then be found as

$$(\boldsymbol{X}(t)|\boldsymbol{\theta}) \sim N\left(\boldsymbol{z}(t) + \boldsymbol{m}(t)\,,\, V(t)\right). \tag{2.12}$$

Simulations using the LNA can be obtained by initialising with $\boldsymbol{z}(0) = \boldsymbol{x}(0), \boldsymbol{m}(0) = \boldsymbol{0}$ and $V(0) = 0$ and recursively drawing from (2.12).

2.5 Example: birth-death model

We now look at one of the most simple types of SKM namely the birth-death model which is used to model population growth and was first introduced in 1925 (Yule et al., 1925).

The birth-death model has just one specie, denoted by X, where x represents the number of individuals in the population as opposed to the number of molecules as previously. As is evident from the name, only two reactions can occur in this model; either a birth or a death. If a birth occurs, the number of individuals in the population increases by one. Conversely, if a death occurs, the number of individuals decreases by one. Using the chemical reaction notation we defined earlier, this model is described by the following system of equations:

$$\begin{aligned} R_1: &\quad X \quad \xrightarrow{\lambda} \quad 2X, \\ R_2: &\quad X \quad \xrightarrow{\mu} \quad \emptyset. \end{aligned}$$

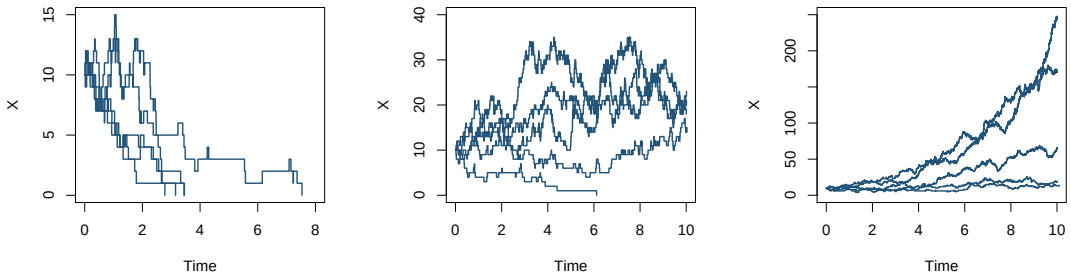

Figure 2.1: Five time course plots of the population levels in the birth-death model with $\mu = 1$, initial condition $x_0 = 10$ and $\lambda = 0.7$ (left), $\lambda = 1$ (middle) and $\lambda = 1.3$ (right).

The corresponding stoichiometry matrix for this system is

$$S = \begin{pmatrix} 1 & -1 \end{pmatrix}$$

with the hazard function

$$h(\boldsymbol{x}(t), \boldsymbol{\theta}) = (\lambda x(t), \mu x(t))^\top,$$

where $\boldsymbol{\theta} = (\log \lambda, \log \mu)^\top$.

Figure 2.1 shows five realisations of the birth-death model for different values of the birth rate λ with the death rate fixed at $\mu = 1$. These are obtained using Gillespie's direct method as detailed in Section 2.4.1 using the hazard functions given above. Each starts with the same initial population size of $x_0 = 10$, where we see the population die out when the birth rate is lower than the death rate ($\lambda < \mu$). When $\lambda = \mu$ we see the population level tends to stabilise, although stochasticity means the population could still die out, which it does for one of the realisations. Finally, we have $\lambda > \mu$ where we see the populations grow rapidly as there is a higher chance of births occuring than deaths.

2.6 Example: Lotka-Volterra model

The Lotka-Volterra model is one of the more well known stochastic kinetic models first introduced by Lotka (1910) and Volterra (1926). Although this model is typically used to describe predator-prey interactions, the theory on stochastic kinetic models that we have developed so far still applies. The model describes the evolution of predator (X_2) and prey (X_1) specie levels through time. It is fairly simple due to a number of assumptions, for example, the level of predator depends entirely on the number of prey. The system is

described by three reactions

$$\begin{aligned} R_1 : \quad & X_1 & \xrightarrow{e^{\theta_1}} \quad & 2X_1, \\ R_2 : \quad & X_1 + X_2 & \xrightarrow{e^{\theta_2}} \quad & 2X_2, \\ R_3 : \quad & X_2 & \xrightarrow{e^{\theta_3}} \quad & \emptyset. \end{aligned}$$

The first reaction denotes a prey birth and the third reaction represents a predator death. The second reaction represents an interaction between the two species which results in a prey death and a predator birth. The corresponding stoichiometry matrix is

$$S = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

with hazard function

$$h(\boldsymbol{x}(t), \boldsymbol{\theta}) = \left(e^{\theta_1} x_1, e^{\theta_2} x_1 x_2, e^{\theta_3} x_2\right)^\top$$

where we have dropped dependence of $\boldsymbol{x}(t)$ on t for notational simplicity. We can again obtain time course plots from this system through the use of Gillespie's direct method where the trajectories exhibit oscillatory behaviour, as depicted in Figure 2.2. The increasing number of prey due to lack of predator eventually results in an increase in predator levels which eventually lead to decreasing levels of prey. Note that the predator specie will die out eventually.

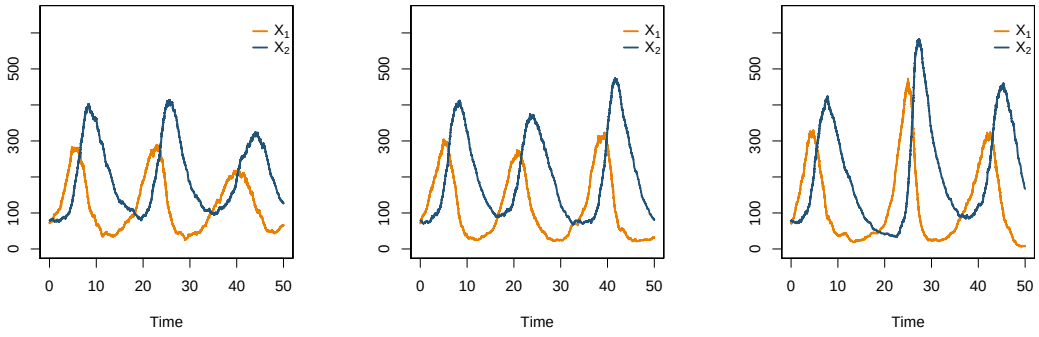

Figure 2.2: Time course plots of three realisations of species in the Lotka-Volterra system with $\boldsymbol{\theta} = \log(0.5, 0.0025, 0.3)^\top$ and $\boldsymbol{x}_0 = (71, 79)^\top$.

2.7 Example: Schlögl system

The Schlögl system is a reaction system consisting of three species, X_1, X_2 and X_3, and four reactions, first proposed by Schlögl (1972). As described in Vellela and Qian (2009),

the system models the concentrations of the dynamic chemical X_1 and the concentrations of chemicals X_2 and X_3. The system is homogeneous in space and describes an exchange of chemicals between two material baths. The chemical reactions are

$$\begin{aligned} R_1: &\quad 2X_1 + X_2 \xrightarrow{e^{\theta_1}} 3X_1, \\ R_2: &\quad 3X_1 \xrightarrow{e^{\theta_2}} 2X_1 + X_2, \\ R_3: &\quad X_3 \xrightarrow{e^{\theta_3}} X_1, \\ R_4: &\quad X_1 \xrightarrow{e^{\theta_4}} X_3. \end{aligned}$$

The corresponding stoichiometry matrix for this system is

$$S = \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

with the hazard function

$$h(\boldsymbol{x}(t), \boldsymbol{\theta}) = \left(\frac{e^{\theta_1} x_1(x_1 - 1)x_2}{2}, \frac{e^{\theta_2} x_1(x_1 - 1)(x_1 - 2)}{6}, e^{\theta_3} x_3, e^{\theta_4} x_1 \right)$$

where again we have dropped dependence of $\boldsymbol{x}(t)$ on t for notational simplicity. The system is particularly interesting due to the bistability it exhibits for particular choices of the rate constants $\boldsymbol{\theta}$ and initial values $\boldsymbol{x}_0$. Modelling this behaviour could be challenging and is one of the reasons we will look at trying to emulate this behaviour. By examining the stoichiometry matrix of this model we see that the system obeys a conservation law, that is $X_1 + X_2 + X_3 = constant$, so we can observe just two of the chemical species, say X_1 and X_2, and calculate the path of X_3 easily if we need to. Figure 2.3 shows the time courses of ten independent realisations of the system for each of the different simulation (or approximation) methods discussed above, each simulated using rate constants $\boldsymbol{\theta} = \log(3 \times 10^{-7}, 10^{-4}, 0.000773, 3.276)^\top$ and initial conditions $\boldsymbol{x}_0 = (250, 10^5, 2 \times 10^5)^\top$. We can see here that Gillespie's direct method and the CLE can both adequately capture the bimodal behaviour whilst the LNA fails to satisfactorily explore both modes.

The bimodality that this system exhibits is one of the reasons we wish to build a fast and flexible emulator where the flexiblity will be key to emulating the system's output. We also note that the Schlögl system has large counts of the second and third species, X_2, X_3. Now imagine if we were to use Gillespie's direct method as part of an inference scheme to learn the stochastic rate constants. The large number of counts means there will also be a large number of reactions occuring in a small space of time since there are so many molecules in the system. Hence the dwell time between each reaction will become

small. Simulating all of these reactions using the Gillespie algorithm would therefore be computationally expensive, especially within an inference scheme requiring many millions of full realisations.

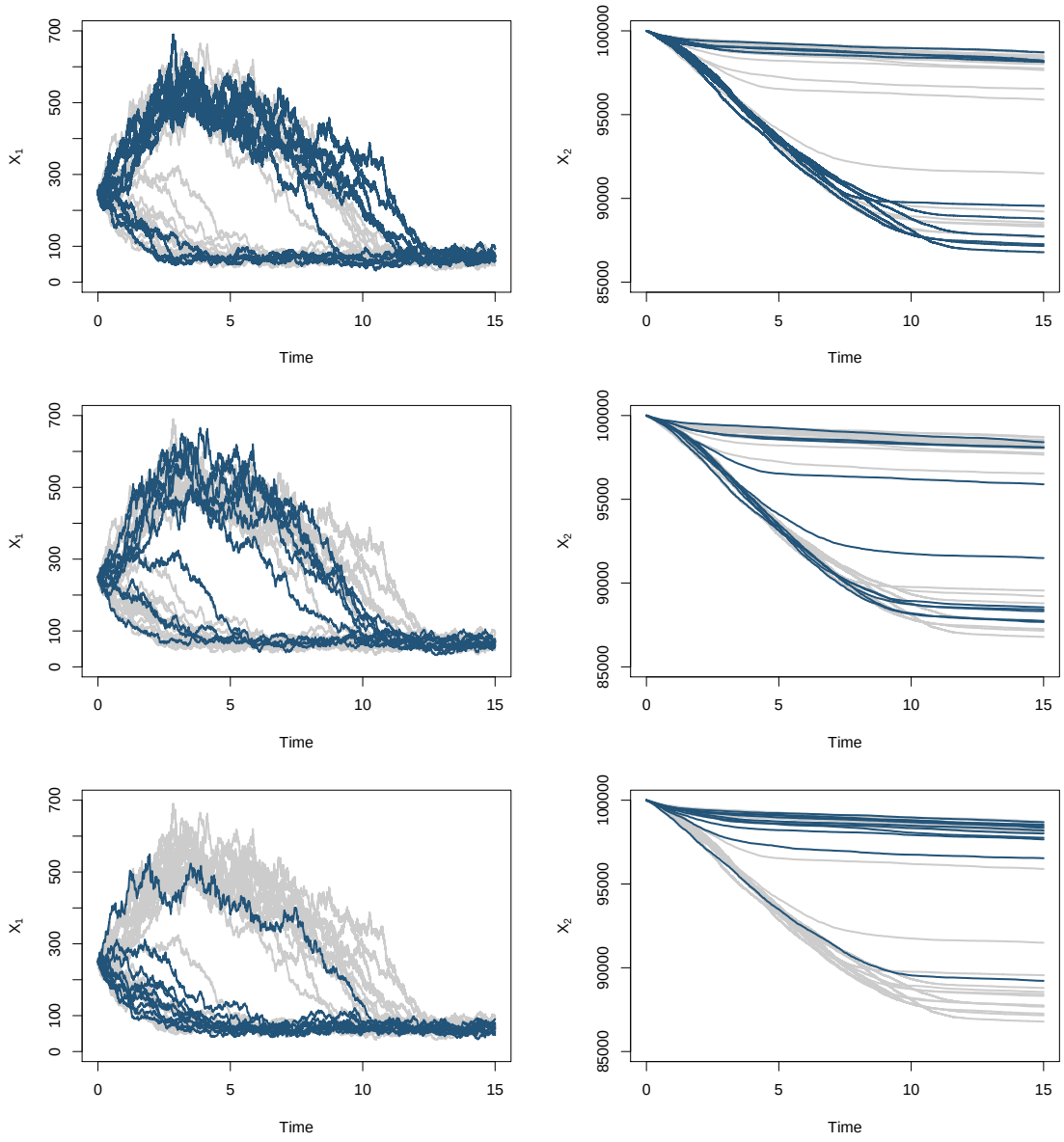

Figure 2.3: Time course plots of the species in the Schlögl system for each of the different simulation methods. All realisations from all methods are given in grey with the top row highlighting (in blue) the ten time courses obtained using Gillespie's direct method, middle row highlighting (in blue) the ten time courses obtained using the CLE and the bottom row highlighting (in blue) the ten time courses obtained using the LNA.

Chapter 3

Bayesian inference

3.1 Introduction

The aim of this thesis is to infer the parameters of a stochastic kinetic model given time course data that may be incomplete and subject to measurement error. We will adopt the Bayesian framework when performing inference and additionally, during the fitting stage of the emulator. A particular issue that arises when using such methods with stochastic kinetic models is that the observed data likelihood is typically intractable, as is the case for the Schlögl system.

This chapter introduces Bayesian inference and outlines a number of methods that can be used to perform inference, including Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) methods.

3.2 Bayes' Theorem

Suppose we have observed data $\boldsymbol{x}$ which can be modelled by a probability density function (or mass function if $\boldsymbol{x}$ is discrete) $f(\boldsymbol{x}|\boldsymbol{\theta})$. The parameters of interest here are the parameters of the density function $\boldsymbol{\theta}$. The aim is then to quantify our uncertainty about $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)^\top$ given our observed quantities $\boldsymbol{x}$. The likelihood function, which contains all of the information from the data, is

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = f(\boldsymbol{x}|\boldsymbol{\theta}).$$

This represents the density (or mass function if $\boldsymbol{x}$ is discrete) of the data $\boldsymbol{x}$ as a function of our unknown quantities $\boldsymbol{\theta}$. Often we have some prior beliefs about these parameters $\boldsymbol{\theta}$, which may be quite strong or weak beliefs, and we represent this prior knowledge through

our prior density (or mass function if our parameters $\boldsymbol{\theta}$ are discrete) $\pi(\boldsymbol{\theta})$. We can then combine our prior beliefs with information gathered from the observed data $\boldsymbol{x}$ through the use of Bayes' theorem to obtain the posterior distribution. Using Bayes' theorem the posterior density is

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x})}{\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x})d\boldsymbol{\theta}}, \tag{3.1}$$

where $\int_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x})d\boldsymbol{\theta}$ is called the marginal likelihood. The marginal likelihood does not depend on $\boldsymbol{\theta}$ and ensures the posterior density integrates to one. Hence we can express Bayes' theorem as

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x}),$$

which shows the posterior is proportional to the prior times the likelihood. When $\boldsymbol{\theta}$ is discrete, we would obtain the posterior mass function using the likelihood and the prior mass function. The marginal likelihood in this case would now be a sum, $\sum_{\boldsymbol{\theta}} \pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x})$. Unless stated otherwise, we shall assume that unknown parameters $\boldsymbol{\theta}$ are continuous.

Obtaining the posterior can be challenging when working with complex models since the marginal likelihood in the denominator of (3.1) will not be available in closed form. If the prior density is of the same functional form as the likelihood, known as conjugacy, then we can find the posterior distribution in closed form. For the majority of cases when examining complex models, conjugate priors are often not available or, if they are, a conjugate prior may not express our prior beliefs sufficiently. In these cases we appeal to other methods of computing the posterior distribution such as Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC).

3.3 Markov chain Monte Carlo (MCMC)

Although approximating the posterior density may be straightforward in simple conjugate cases, models of interest are often rather complex or high dimensional, making it difficult to obtain the normalising constant. Markov chain Monte Carlo (MCMC) is a method widely used when conducting Bayesian inference that allows us, in principle, to generate samples from complex and high-dimensional posterior distributions. A large area of research exists on MCMC and we direct the reader to Brooks et al. (2011), and Gamerman and Lopes (2006) for reviews.

The approach allows us to simulate from a Markov chain whose stationary distribution is our target distribution of interest, in our case the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x})$. We can start the chain at an initial point of our choosing in the support of the posterior and after a sufficient amount of iterations, the chain will eventually converge to the posterior, subject to certain conditions, for example, ergodicity and detailed balance. Once the chain

has converged, any samples obtained will be realisations from the posterior distribution. Two methods that can be used to construct such chains are the Gibbs sampler and the Metropolis-Hastings algorithm which we now discuss.

3.3.1 Gibbs sampling

The Gibbs sampler was first introduced by Geman and Geman (1984) and has since become a common tool when constructing MCMC algorithms.

Suppose that the posterior distribution, $\pi(\boldsymbol{\theta}|\boldsymbol{x})$, is difficult or even impossible to directly sample from but the conditional distributions of each of the components of $\boldsymbol{\theta}$ are available for sampling from. Gibbs sampling makes use of these full conditional distributions (FCDs), denoted by $\pi(\theta_i|\boldsymbol{\theta}_{-i}, \boldsymbol{x})$, by sampling from each of them in turn. After a suitable amount of iterations of the sampler, often referred to as burn-in, the chain converges to the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{x})$. A justification of this 'fixed sweep' Gibbs sampler can be found in Gamerman and Lopes (2006). The Gibbs sampler is described in Algorithm 2.

Algorithm 2 Gibbs sampler

1. Initialise the state of the chain to $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_d^{(0)})^\top$ and set the iteration counter to $j = 1$.
2. Obtain a new sample $\boldsymbol{\theta}^{(j)}$ from $\boldsymbol{\theta}^{(j-1)}$ by successive generation from the full conditional distributions

$$\begin{aligned}
\theta_1^{(j)} &\sim \pi(\theta_1|\theta_2^{(j-1)}, \ldots, \theta_d^{(j-1)}, \boldsymbol{x}) \\
\theta_2^{(j)} &\sim \pi(\theta_2|\theta_1^{(j)}, \theta_3^{(j-1)}, \ldots, \theta_d^{(j-1)}, \boldsymbol{x}) \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
\theta_d^{(j)} &\sim \pi(\theta_d|\theta_1^{(j)}, \ldots, \theta_{d-1}^{(j)}, \boldsymbol{x})
\end{aligned}$$

3. Set $j := j + 1$ and return to step 2.

3.3.2 Metropolis-Hastings

In scenarios where the full conditional distributions are not easy to sample from, an alternative to Gibbs sampling is required, and so we turn to Metropolis-Hastings sampling.

The algorithm was introduced by Metropolis et al. (1953) and followed up by Hastings (1970). The method makes use of a proposal density, $q(\cdot|\cdot)$, which should have the same support as the target and should be easy to simulate from. A proposed value of $\boldsymbol{\theta}$ is

obtained from the associated proposal distribution and then accepted or rejected according to the acceptance probability which depends on the target distribution, in our case the posterior. The Metropolis-Hastings algorithm is given by Algorithm 3.

We see that the posterior density appears as a ratio in the acceptance probability and hence is only needed up to a constant of proportionality.

Algorithm 3 Metropolis-Hastings sampler

1. Initialise the state of the chain to $\boldsymbol{\theta}^{(0)}$ and set the iteration counter to $j = 1$.
2. Generate a proposed value $\boldsymbol{\theta}^*$ from the proposal distribution $q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(j-1)})$.
3. Calculate the acceptance probability $\alpha(\boldsymbol{\theta}^{(j-1)}, \boldsymbol{\theta}^*)$ where
$$\begin{aligned}\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}^*) &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*|\boldsymbol{x})q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}|\boldsymbol{x})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\} \\ &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)L(\boldsymbol{\theta}^*|\boldsymbol{x})q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^*)L(\boldsymbol{\theta}^*|\boldsymbol{x})q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}\right\}\end{aligned}$$
4. With probability $\alpha(\boldsymbol{\theta}^{(j-1)}, \boldsymbol{\theta}^*)$ set $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^*$ and set $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j-1)}$ otherwise.
5. Set $j := j + 1$ and return to step 2.

It should be clear that Algorithm 3 defines a homogenous Markov chain.

The proposal density plays an important role in the mixing efficiency of the Markov chain. Ideally, the Markov chain should move rapidly over the entire parameter space. We now look at typical choices of $q(\cdot|\cdot)$ when constructing a Metropolis-Hastings scheme.

Symmetric proposal

If the proposal distribution is chosen to be symmetric, the acceptance probability simplifies. A symmetric proposal density has $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)$, $\forall\, \boldsymbol{\theta}^*, \boldsymbol{\theta}$. Through cancellation this gives

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*|\boldsymbol{x})}{\pi(\boldsymbol{\theta}|\boldsymbol{x})}\right\}.$$

Hence the acceptance probability does not depend on the proposal density.

Random walk proposal

Here, the proposal mechanism is a random walk so that, at iteration j,

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(j-1)} + \boldsymbol{w}_j$$

where $\boldsymbol{w}_j$ are independent and identically distributed random variables typically chosen to be normally distributed with a mean of $\mathbf{0}$.

Of course, when choosing a distribution for $\boldsymbol{w}_j$, we need to determine an appropriate variance. If we set the variance too small then the majority of moves will be accepted since they will constitute small jumps in the parameter space, leading to a chain that is exploring the space too slowly. Setting the variance too high and we risk too many moves being rejected leading to poor mixing of the chain and poor exploration of the parameter space.

Roberts et al. (2001) show that an acceptance probability of 0.234 is optimal, subject to assumptions regarding the target density and as $d \to \infty$, where d is the dimension of $\boldsymbol{\theta}$. This leads to the heuristic of setting

$$\text{Var}(\boldsymbol{w}_j) = \frac{2.38^2}{d}\widehat{\text{Var}}(\boldsymbol{\theta}|\boldsymbol{x}).$$

Independence chains

If the proposal distribution is constructed independently of the current position of the chain we obtain an independence chain (Tierney, 1994). The proposal density is then $q(\boldsymbol{\theta}^*|\boldsymbol{\theta}) = g(\boldsymbol{\theta}^*)$. The acceptance probability in this case is

$$\begin{aligned}\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}^*) &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*|\boldsymbol{x})}{\pi(\boldsymbol{\theta}|\boldsymbol{x})}\frac{g(\boldsymbol{\theta})}{g(\boldsymbol{\theta}^*)}\right\} \\ &= \min\left\{1, \frac{\pi(\boldsymbol{\theta}^*)L(\boldsymbol{\theta}^*|\boldsymbol{x})}{\pi(\boldsymbol{\theta})L(\boldsymbol{\theta}|\boldsymbol{x})}\frac{g(\boldsymbol{\theta})}{g(\boldsymbol{\theta}^*)}\right\}.\end{aligned}$$

The acceptance probability is increased by choosing $g(\cdot)$ to be close to $\pi(\cdot|\boldsymbol{x})$. One possibility is to take the proposal density to be the prior density in which case the acceptance probability simplifies down to a ratio of likelihoods evaluated at the current and proposed parameter value. That is

$$\alpha(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \min\left\{1, \frac{f(\boldsymbol{x}|\boldsymbol{\theta}^*)}{f(\boldsymbol{x}|\boldsymbol{\theta})}\right\}.$$

Hybrid methods

Consider a scenario where full conditional distributions (FCDs) are available for a subset of components of $\boldsymbol{\theta}$. It is then possible to perform a Gibbs update for those variables for which we have tractable FCDs and then perform Metropolis-Hastings steps for the rest.

The resulting chain is termed a hybrid chain. The updates can be performed componentwise as in the Gibbs sampling algorithm as opposed to a block update of $\boldsymbol{\theta}$. This is then referred to as Metropolis-within-Gibbs (Gamerman and Lopes, 2006). Of course, we could perform componentwise Metropolis-Hastings steps for all components of $\boldsymbol{\theta}$ if the FCDs weren't easy to sample from.

It can be shown that the Gibbs sampler is a special case of the Metropolis-Hastings algorithm, with each iteration consisting of d Metropolis-Hastings steps targeting $\pi(\theta_i|\boldsymbol{\theta}_{-i}, \boldsymbol{x})$ with proposal density $\pi(\theta_i|\boldsymbol{\theta}_{-i}, \boldsymbol{x})$.

3.3.3 Analysis of MCMC output

When performing MCMC methods to sample from the posterior distribution of interest, we need to monitor convergence. Starting from an initial value, the chain will have a number of iterates that are known as *burn-in* before the chain has converged. These iterates should be removed. Graphical summaries of the iterates can be used to informally check convergence. A common technique is to view trace plots of the iterates and check the chain is mixing well and for any obvious problems in the chain, for example poor mixing or the chain sticking at a number of modes. Autocorrelation plots can also be used to monitor the autocorrelation in the chain.

There exist more formal methods for checking convergence, see for example Gelman et al. (1992). The authors here suggest running multiple chains with different initial values and checking that the chains are indistinguishable after some time, after all, the initial value should not matter if the chain is correctly sampling from the posterior distribution.

If autocorrelation is high after checking autocorrelation plots, it is possible to *thin* the output to reduce autocorrelation. Thinning involves keeping every i^{th} iteration and discarding others to ensure that the resulting samples exhibit low autocorrelation. This technique is also useful for reducing the storage costs associated with particularly long runs. Some insight in to how to choose the burn-in period and level of thinning can be found in Raftery and Lewis (1992).

Once we are satisfied that the chain has converged and is correctly sampling from the posterior distribution we can analyse the output in a number of ways. Plotting histograms or density plots of the chain allows us to see the shape of the posterior marginal distributions.

Calculating summary statistics of the output gives us estimates of the summary statistics of the posterior distribution, for example, the expectation of a particular random variable can be obtained via

$$\mathrm{E}[\theta_i|\boldsymbol{x}] \approx \frac{1}{N}\sum_{j=1}^{N}\theta_i^{(j)}$$

where N is the total number of samples obtained from the MCMC run (after burn-in and thinned samples are removed) and $\theta_i^{(j)}$ is the j^{th} sample from the posterior distribution for θ_i.

3.4 Likelihood-free methods

We now discuss the scenario when the likelihood is intractable which is typical for stochastic kinetic models as described in Chapter 2. In this case, it is not possible to evaluate the likelihood however it is possible to simulate from the stochastic kinetic model of interest as described in Section 2.4. Likelihood-free methods of inference allow for us to overcome evaluating the likelihood by utilising the ability to simulate from the model instead. To better elucidate the applicability of likelihood-free methods for SKMs, we consider a state-space framework, within which it is possible to cast an SKM.

3.4.1 State-space framework

Inferring the rate constants of stochastic kinetic models outlined in Chapter 2 using observations of the system at discrete times is challenging. Suppose that the chemical reaction system is observed with error at times $t = 1, \ldots, T$ according to some observation error model with parameters Σ. Let $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$ denote this collection of observed data and $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ represent the latent states at the observation times. Both the latent states and the observations will depend on parameters $\boldsymbol{\theta}$ and the state at time t will depend only on the state at time $t-1$, that is the process satisfies the Markov property

$$\pi(\boldsymbol{x}_t|\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{\theta}) = \pi(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta}),$$

where $\pi(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{\theta})$ is the transition density of the state-space model (or the mass function of an MJP), which we assume is easy to simulate from. The observations $\{\boldsymbol{y}_t\}$ are conditionally independent given $\{\boldsymbol{x}_t\}$ and $\boldsymbol{\theta}$.

Bayesian inference proceeds through the posterior density

$$\pi(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\pi(\Sigma)\pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma) \tag{3.2}$$

where $\pi(\boldsymbol{\theta})$ is the prior density for the log kinetic rates $\boldsymbol{\theta}$, $\pi(\Sigma)$ is the prior density for the error model parameters Σ, $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ is the probability of the unobserved stochastic process and $\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma)$ is the observation error density which we assume can be evaluated. Note that this factorisation of the prior density for $(\boldsymbol{\theta}, \Sigma)^\top$ is based on the assumption that $\boldsymbol{\theta}$ and Σ are independent *a priori*. Unfortunately the posterior density $\pi(\boldsymbol{\theta}, \boldsymbol{x}, \Sigma|\boldsymbol{y})$ is typically unavailable in closed form and so samples from it are usually generated through a suitable MCMC scheme, which we consider in the next section.

We will discuss a scheme with a proposal inspired by approximate Bayesian computation (ABC), that is, one in which system data $\boldsymbol{x}$ are simulated from the stochastic process $\mathcal{S}(\boldsymbol{\theta})$ using Gillespie's direct method, see Section 2.4.1. Approximate Bayesian computation (ABC) is an example of likelihood-free inference where the general idea is to simulate many datasets from the model for different values of our parameters and compare the simulated data to the observed data. If the simulated data are close enough to the observed data, subject to some criteria, then the proposed value of the parameter is accepted. It is typical to compare summary statistics of the simulated data with summary statistics of the observed data and if the difference is within some tolerance ϵ then the sample will be accepted. It is important to note that ABC methods do not generate samples from the true posterior but rather an approximate posterior, hence the name. For a general background of ABC methods, see Marjoram et al. (2003) with applications in bioinformatics found in Wilkinson (2018), Liepe et al. (2014), Liepe et al. (2010) and Toni et al. (2008).

3.4.2 Likelihood-free MCMC

Consider again the setting described above with noisy data $\boldsymbol{y}$ and the task of inferring the rate constants $\boldsymbol{\theta}$. Given the complexity of (3.2), most notably the dimension of $\boldsymbol{x}$ and the complex likelihood structure, we consider a method that avoids some of this complexity by exploiting the ability to forward simulate from $\pi(\boldsymbol{x}|\boldsymbol{\theta})$.

Consider first the case of fixed and known Σ. A Metropolis-Hastings scheme is constructed via a joint update of $\boldsymbol{\theta}$ and $\boldsymbol{x}$. Suppose the current state of the Markov chain is $(\boldsymbol{\theta}, \boldsymbol{x})$. A proposed value, $\boldsymbol{\theta}^*$, is drawn from $q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$. Then, conditional on this value, a proposed new sample path $\boldsymbol{x}^*$ is drawn from $\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)$ via Gillespie's direct method (Algorithm 1). Together, the newly proposed pair is accepted with probability $\min\{1, A\}$ where

$$\begin{aligned} A =& \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{y}|\boldsymbol{x}^*, \Sigma)}{\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma)} \frac{\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)}{\pi(\boldsymbol{x}|\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{x}|\boldsymbol{\theta})}{\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)} \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})} \\ =& \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{y}|\boldsymbol{x}^*, \Sigma)}{\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma)} \times \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})}. \end{aligned}$$

Note that the problematic term $\pi(\boldsymbol{x}|\boldsymbol{\theta})$ cancels in the acceptance probability, since it ap-

pears in both the target and (joint) proposal densities. Such Metropolis-Hastings schemes are typically referred to as 'likelihood-free' (Marjoram et al., 2003) and are closely related to approximate Bayesian computation (ABC) approaches (Beaumont et al., 2002).

It remains that we can update Σ. As is often assumed in practice, we take an observation model corresponding to additive Gaussian noise and Σ as a diagonal matrix with dimension given by the number of observed components with σ_i^2 in the i^{th} diagonal entry. In this case, under the specification of a semi-conjugate prior, Gibbs steps are possible for each σ_i^2. The likelihood-free Metropolis-Hastings scheme is given by Algorithm 4.

In practice, this particular Metropolis-Hastings scheme can suffer from small acceptance rates, unless the observed time course is relatively short (e.g. $T \leq 50$). An alternative approach is to update our beliefs as each observation becomes available, which we now discuss.

3.4.3 Sequential Monte Carlo (SMC)

Sequential Monte Carlo (SMC) (Smith, 2013) methods allow for sampling the sequence of target densities $\pi(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ for $t = 1, 2, \ldots, T$, with $\boldsymbol{y}_{1:t} = (y_1, \ldots, y_t)^\top$. Consider first the case of known $\boldsymbol{\theta}$ and Σ which we drop from the notation. Suppose that, given the posterior $\pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, we observe $\boldsymbol{y}_{t+1}$ at time $t+1$. Applying Bayes' theorem sequentially gives

$$\pi(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t|\boldsymbol{y}_{1:t+1}) \propto \pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})\pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)\pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1})$$

and therefore

$$\begin{aligned}\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1}) &\propto \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}) \int \pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})\pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)\,\mathrm{d}\boldsymbol{x}_t \\ &\propto \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1})\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t}).\end{aligned} \tag{3.3}$$

Of course, $\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t})$ will be intractable in practice. Inference may still proceed if a sample from $\pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ is available, by using a suitable sampling based approach. Within the context of SMC, this approach is known as importance resampling (sometimes referred to as weighted resampling).

Importance resampling

We consider first a related Monte Carlo integration technique known as importance sampling. Suppose we have an integral of the form

$$\begin{aligned} I &= \int f(\boldsymbol{x}) g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \\ &= \mathrm{E}(f(\boldsymbol{X})) \end{aligned}$$

where $g(\boldsymbol{x})$ is a probability density function and evaluation of I is difficult. If it is possible to generate (independent) draws $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ from $g(\cdot)$ then

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}^{(i)})$$

is an unbiased estimator of I. If sampling from $g(\cdot)$ is difficult but we can find an alternative 'proposal' density $q(\cdot)$ with the same support as $g(\cdot)$ then we first rewrite

$$\begin{aligned} I &= \int f(\boldsymbol{x}) g(\boldsymbol{x}) \frac{q(\boldsymbol{x})}{q(\boldsymbol{x})} \, \mathrm{d}\boldsymbol{x} \\ &= \mathrm{E}\left(\frac{f(\boldsymbol{x}) g(\boldsymbol{x})}{q(\boldsymbol{x})} \right) \end{aligned}$$

where the expectation is with respect to $q(\cdot)$. Hence, given draws $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ from $q(\cdot)$,

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\boldsymbol{x}^{(i)}) g(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}$$

gives an unbiased estimate of I. This method is known as importance sampling.

Note that by the strong law of large numbers

$$\int \frac{g(\boldsymbol{x})}{q(\boldsymbol{x})} q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \simeq \frac{1}{N} \sum_{i=1}^{N} \frac{g(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})} \to 1 \text{ as } n \to \infty.$$

This leads to the asymptotically unbiased self-normalised importance sampling estimate

$$\begin{aligned} \hat{I} &= \frac{\frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}^{(i)}) [g(\boldsymbol{x}^{(i)}) / q(\boldsymbol{x}^{(i)})]}{\frac{1}{N} \sum_{i=1}^{N} [g(\boldsymbol{x}^{(i)}) / q(\boldsymbol{x}^{(i)})]} \\ &= \sum_{i=1}^{N} f(\boldsymbol{x}^{(i)}) w^{(i)} \end{aligned} \tag{3.4}$$

where

$$w^{(i)} = \frac{g(\boldsymbol{x}^{(i)})/q(\boldsymbol{x}^{(i)})}{\sum_{j=1}^{N} g(\boldsymbol{x}^{(j)})/q(\boldsymbol{x}^{(j)})}, \quad i = 1, \ldots, N$$

are known as normalised importance weights.

Intuitively from (3.4), the normalised weights $w^{(i)}$ can be thought of as the probability associated with the discrete sample $\boldsymbol{x}^{(i)}$. In importance resampling, we resample (with replacement) N times amongst the set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ using the weights as probabilities. The resulting sample is approximately distributed according to $g(\cdot)$, and the algorithm is exact as $N \to \infty$ (Smith and Gelfand, 1992).

Bootstrap filter

Consider again the target density in (3.3). Suppose that we have an equally weighted sample $\{\boldsymbol{x}_t^{(i)}\}, i = 1, \ldots, N$ from $\pi(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$. SMC methods adopt the approximation

$$\hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}).$$

We see that the continuous support of $\boldsymbol{x}_t$ is replaced by the discrete support of the particle set, allowing the integral in (3.3) to be replaced by a sum. Therefore we obtain

$$\hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1}) \propto \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}) \sum_{i=1}^{N} \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}). \tag{3.5}$$

We sample $\hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1})$ by applying importance resampling with a proposal density $q(\boldsymbol{x}_{t+1}) = \hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t})$. That is, for each $\{\boldsymbol{x}_t^{(i)}\}$, we draw $\tilde{\boldsymbol{x}}_{t+1}^{(i)} \sim \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)})$ using Gillespie's direct method and assign the normalised weight

$$w_{t+1}^{(i)} = \frac{\hat{w}_{t+1}^{(i)}}{\sum_{j=1}^{N} \hat{w}_{t+1}^{(j)}}, \quad i = 1, \ldots, N$$

where

$$\hat{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}), \quad i = 1, \ldots, N.$$

Resampling (with replacement) N times amongst $\{\tilde{\boldsymbol{x}}_{t+1}^{(1)}, \ldots, \tilde{\boldsymbol{x}}_{t+1}^{(N)}\}$ using the weights as probabilities, gives an equally weighted sample $\{\boldsymbol{x}_{t+1}^{(1)}, \ldots, \boldsymbol{x}_{t+1}^{(N)}\}$ approximately distributed according to $\pi(\boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1})$.

After initialising with an equally weighted sample $\{\boldsymbol{x}_0^{(1)}, \ldots, \boldsymbol{x}_0^{(N)}\}$ from the prior $\pi(\boldsymbol{x}_0)$, we apply this technique recursively for $t = 1, 2, \ldots, T$. The resulting algorithm typically referred to as the bootstrap filter (Gordon et al. (1993) and Cappé et al. (2007)) or

sequential importance resampling (SIR).

Extending this approach to include the static parameters is straightforward in principle. Suppose that we have an equally weighted sample $\{(\boldsymbol{\theta}^{(1)}, \Sigma^{(1)}, \boldsymbol{x}_t^{(1)}), \ldots, (\boldsymbol{\theta}^{(N)}, \Sigma^{(N)}, \boldsymbol{x}_t^{(N)})\}$ from $\pi(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}_t|\boldsymbol{y}_{1:t})$. Rewriting the target in (3.5) to include $\boldsymbol{x}_t$ gives

$$\pi(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}_t, \boldsymbol{x}_{t+1}|\boldsymbol{y}_{1:t+1}) \propto \sum_{i=1}^{N} \delta_{(\boldsymbol{\theta}^{(i)}, \Sigma^{(i)}, \boldsymbol{x}_t^{(i)})} \times \pi(\boldsymbol{y}_{t+1}|\boldsymbol{x}_{t+1}, \Sigma^{(i)}) \times \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$$

where $\delta_{(\boldsymbol{\theta}^{(i)}, \Sigma^{(i)}, \boldsymbol{x}_t^{(i)})}$ is the Dirac mass function that assigns 1 if $(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}_t) = (\boldsymbol{\theta}^{(i)}, \Sigma^{(i)}, \boldsymbol{x}_t^{(i)})$ and 0 otherwise. The SIR filter then proceeds by sampling $\tilde{\boldsymbol{x}}_{t+1}^{(i)} \sim \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ for each $\boldsymbol{x}_t^{(i)}$ and $\boldsymbol{\theta}^{(i)}$. The unnormalised weight is

$$\tilde{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma^{(i)}), \quad i = 1, \ldots, N.$$

The full SIR filter is described in Algorithm 5. In what follows, we refer to the number of particles as N_p.

Unfortunately, SMC methods for static parameters are well known to be afflicted by sample impoverishment (Li et al., 2014). This occurs when only a small minority of parameter particles have reasonable weight. After resampling, only a few distinct parameter particles remain. A number of ad-hoc approaches have been proposed to alleviate this issue. For example, Gordon et al. (1993) (see also Liu and West (2001)).

A key issue with both SMC and MCMC schemes that we've discussed is that they can be both computationally intensive and time consuming due to the constant need to simulate proposed realisations from the stochastic process at each iteration using Gillespie's direct method for example. This problem can become insurmountable if the number of species or reactions are not small (see the Schlögl system in Section 2.7) as the time and resources needed could become unmanageable. A common way around this problem is to simulate the stochastic realisations (or simply just their values at the observed time-points) using a fast, efficient but accurate approximation to the stochastic process. Such approximations are often called *emulators* which we shall now move on to look at constructing.

Algorithm 4 Likelihood-free MCMC algorithm

1. Initialise. Set $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{x}^{(0)}$, e.g. $\boldsymbol{\theta}^{(0)} \sim \pi(\boldsymbol{\theta})$ and set $\boldsymbol{x}^{(0)}$ to be an appropriate value. Put $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)}$ and $\boldsymbol{x} = \boldsymbol{x}^{(0)}$. Set $j = 0$.

2. Generate a new candidate $\boldsymbol{\theta}^*$ from the symmetric proposal distribution $q(\boldsymbol{\theta}^*|\boldsymbol{\theta})$.

3. Propose $\boldsymbol{x}^* \sim \pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)$.

4. Construct the acceptance probability $\min(1, A)$ where

$$\begin{aligned} A &= \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{x}^*, \boldsymbol{y}|\boldsymbol{\theta}^*, \Sigma)}{\pi(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}, \Sigma)} \times \frac{q(\boldsymbol{\theta}|\boldsymbol{\theta}^*)\pi(\boldsymbol{x}|\boldsymbol{\theta})}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta})\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)} \\ &= \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{x}^*, \boldsymbol{y}|\boldsymbol{\theta}^*, \Sigma)}{\pi(\boldsymbol{x}^*|\boldsymbol{\theta}^*)} \times \frac{\pi(\boldsymbol{x}|\boldsymbol{\theta})}{\pi(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}, \Sigma)} \\ &= \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \times \frac{\pi(\boldsymbol{y}|\boldsymbol{x}^*, \Sigma)}{\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma)} \\ &= \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta})} \prod_{t=1}^{T} \frac{\pi(\boldsymbol{y}_t|\boldsymbol{x}_t^*, \Sigma)}{\pi(\boldsymbol{y}_t|\boldsymbol{x}_t, \Sigma)}, \end{aligned}$$

   with probability $\min(1, A)$ put $(\boldsymbol{\theta}^{(j)}, \boldsymbol{x}^{(j)}) = (\boldsymbol{\theta}^*, \boldsymbol{x}^*)$ and $(\boldsymbol{\theta}, \boldsymbol{x}) = (\boldsymbol{\theta}^*, \boldsymbol{x}^*)$, otherwise put $(\boldsymbol{\theta}^{(j)}, \boldsymbol{x}^{(j)}) = (\boldsymbol{\theta}, \boldsymbol{x})$.

5. Draw Σ from $\pi(\Sigma|\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{y})$ and put $\Sigma^{(j)} = \Sigma$.

6. Set $j = j + 1$ and go to step 2.

Algorithm 5 Bootstrap filter

- Initialise
  At time $t = 0$, for $i = 1, \ldots, N_p$
  1. Sample $\boldsymbol{\theta}^{(i)} \sim \pi(\boldsymbol{\theta})$.
  2. Sample $\boldsymbol{X}_t^{(i)} \sim \pi(\boldsymbol{x}_t)$ and $\tilde{\boldsymbol{X}}_{t+1}^{(i)} \sim \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ using Gillespie's direct method.
  3. Calculate and assign a weight to each particle $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)})$ where $\tilde{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma^{(i)})$.
  4. Resample $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma^{(i)})$ N_p times with replacement according to normalised weights $\{w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N_p)}\}$.
- For times $t = 1, \ldots, T$ and $i = 1, \ldots, N_p$
  1. Sample $\tilde{\boldsymbol{X}}_{t+1}^{(i)} \sim \pi(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t^{(i)}, \boldsymbol{\theta}^{(i)})$ using Gillespie's direct method.
  2. Calculate and assign a weight to each particle $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma^{(i)})$ where $\tilde{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma^{(i)})$.
  3. Resample $(\boldsymbol{\theta}^{(i)}, \boldsymbol{x}_t^{(i)}, \Sigma^{(i)})$ N_p times according to the normalised weights $\{w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N_p)}\}$.

Chapter 4

Gaussian processes

4.1 Introduction

The aim of this thesis is to infer the posterior distributions of the rate constants of complex stochastic kinetic models, in particular the Schlögl system which was described in Chapter 2.

Chapter 3 introduced a number of algorithms that can be used to draw samples from the posterior distribution including two likelihood-free methods. These methods avoid having to evaluate the intractable observed data likelihood but do require the ability to simulate from the model for a proposed set of rate constants which would then either be accepted or rejected. One possibility is to use Gillespie's direct method to simulate from the model but this would require simulating a very large number of events which would be quite time consuming for models such as the Schlögl system. The computational cost can be alleviated somewhat by replacing the expensive simulator with a cheap surrogate.

In this chapter we begin to consider the emulator we are building. First we give an overview of Gaussian processes and explain how they can be used for regression, since the emulation task can be seen as a regression problem. We will describe how we can train a Gaussian process given a set of inputs and outputs, and finally, how the fitted GP can be used for prediction.

4.2 GP Regression

Consider the task of inference for input-output mappings using empirical data. We will assume continuous output data so that the problem is one of regression. Denote an input as $\boldsymbol{\theta}$ and an output as $x = x(\boldsymbol{\theta})$. Suppose that we have data $\mathcal{D} = \{(\boldsymbol{\theta}_i, x_i), i = 1, \ldots, N\}$.

Our primary goal is to make predictions for new inputs $\boldsymbol{\theta}^*$. Clearly we need a function $X(\boldsymbol{\theta})$ that allows prediction for all possible input values. One possibility is to assume some parametric form for X and limit attention to that class of functions (for example, linear functions of the input). Another approach is to give prior probability to every possible function, giving higher probability to functions we consider more likely. Although this latter approach may seem intractable, since we have an uncountably infinite set of functions, it turns out to be possible through the use of a Gaussian process.

Informally, a GP can be thought of as the extension of the multivariate Gaussian distribution to infinite dimension and are thus considered as non-parametric models. Rasmussen and Williams (2005) give a formal definition; a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. Just as a Gaussian distribution is completely specified by its mean and variance, a GP is completely specified by its mean function and covariance function.

Consider a real process $X(\boldsymbol{\theta})$ and let $m(\boldsymbol{\theta})$ be the mean function and $k(\boldsymbol{\theta}, \boldsymbol{\theta'})$ be the covariance function such that

$$m(\boldsymbol{\theta}) = \mathrm{E}(X(\boldsymbol{\theta}))$$

and

$$\begin{aligned} k(\boldsymbol{\theta}, \boldsymbol{\theta'}) &= Cov(X(\boldsymbol{\theta}), X(\boldsymbol{\theta'})) \\ &= \mathrm{E}\left[(X(\boldsymbol{\theta}) - m(\boldsymbol{\theta}))(X(\boldsymbol{\theta'}) - m(\boldsymbol{\theta'}))\right]. \end{aligned}$$

We write the GP prior as

$$X(\boldsymbol{\theta}) \sim GP(m(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta'})),$$

where $X(\boldsymbol{\theta})$ represents the value of the function at location $\boldsymbol{\theta}$.

Suppose we wish to characterise the GP at a finite set of inputs. Set

$$\boldsymbol{\Theta}_N = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N)^\top$$

where, for example, $\boldsymbol{\theta}_1 = (\theta_{1,1}, \ldots, \theta_{1,d})^\top$ and

$$\begin{aligned} \boldsymbol{X} &= X(\boldsymbol{\Theta}_N) \\ &= (X(\boldsymbol{\theta}_1), \ldots, X(\boldsymbol{\theta}_N))^\top. \end{aligned}$$

Using the definition of a GP, we know that $\boldsymbol{X}$ has a Gaussian distribution with a mean vector $m(\boldsymbol{\Theta}_N)$ whose i^{th} element is $m(\boldsymbol{\theta}_i)$, and variance matrix $K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)$ whose $(i,j)^{th}$

element is $k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$. That is

$$\boldsymbol{X} \sim N(m(\boldsymbol{\Theta}_N), K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)) \tag{4.1}$$

where

$$m(\boldsymbol{\Theta}_N) = (m(\boldsymbol{\theta}_1), \ldots, m(\boldsymbol{\theta}_N))^\top$$

and

$$K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N) = \begin{pmatrix} k(\boldsymbol{\theta}_1, \boldsymbol{\theta}_1) & k(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) & \cdots & k(\boldsymbol{\theta}_1, \boldsymbol{\theta}_N) \\ k(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1) & k(\boldsymbol{\theta}_2, \boldsymbol{\theta}_2) & \cdots & k(\boldsymbol{\theta}_2, \boldsymbol{\theta}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{\theta}_N, \boldsymbol{\theta}_1) & k(\boldsymbol{\theta}_N, \boldsymbol{\theta}_2) & \cdots & k(\boldsymbol{\theta}_N, \boldsymbol{\theta}_N) \end{pmatrix}. \tag{4.2}$$

The only constraint on $k(\cdot, \cdot)$ is that it must generate a symmetric, invertible, positive definite covariance matrix for any $\boldsymbol{\Theta}_N$. We now consider choices of $m(\cdot)$ and $k(\cdot, \cdot)$.

4.2.1 Mean function

GP prio A typical choice for $m(\cdot)$ is to use a contant mean function, usually 0, which denotes a lack of prior knowledge of the output. Choosing an appropriate mean function when using a GP for emulation is important since the fitted mean will revert back to the prior mean function when we are far away from any training points. We could choose a fixed mean function if we were confident about the mean of the process. However, it is often more convenient to specify a basis function whose coefficients $\boldsymbol{\beta}$ can be inferred from the data. Bastos and OHagan (2009) suggest this basis function should be chosen to incorporate any expert prior belief about the underlying function that we are trying to model with the GP. One choice of basis function is a linear combination of the inputs, that is

$$m(\boldsymbol{\theta}) = \beta_0 + \sum_{l=1}^{d} \beta_l \theta_l, \tag{4.3}$$

which expresses the data being close to a linear model.

4.2.2 Covariance function

The covariance function specifies the covariance between the outputs and is a function of the input points $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. The choice of covariance function will determine the properties of $X(\boldsymbol{\theta})$ (e.g. stationarity, smoothness, periodicity etc.). Additionally, we would expect output to be similar for two points that are close in $\boldsymbol{\theta}$ space and hence we need to choose a covariance function that has this property.

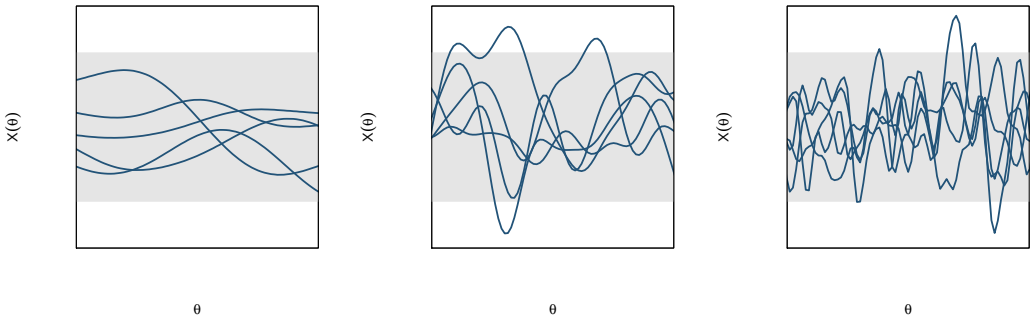

Figure 4.1: Random draws from a GP with squared exponential covariance function and hyperparameters $r = 1$ (left), $r = 0.1$ (middle) and $r = 0.01$ (right) with $\sigma^2 = 1$. Grey bands represent 95% probability intervals.

A typical choice of covariance function is the squared exponential covariance function (O'Hagan (2006), Rasmussen and Williams (2005)) which depends on hyperparameters σ^2 and $\boldsymbol{r}$ with the form

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma^2 \exp\left\{ -\sum_{l=1}^{d} \frac{(\theta_l - \theta_l')^2}{r_l^2} \right\}. \tag{4.4}$$

Alternative covariance functions include the Matérn and rational quadratic covariance function to name a few. Note that the squared exponential covariance function is a special case of the Matérn covariance function. The squared exponential has desirable properties such as stationarity and being infinitely differentiable so that GP realisations are always smooth.

We now look at the effect of changing the parameters of the squared exponential covariance function. We can see this by drawing realisations from a zero mean GP with covariance function as in (4.4), as seen in Figures 4.1 and 4.2.

Figure 4.1 shows the effect of changing the correlation length parameter $\boldsymbol{r}$. In this illustration $\boldsymbol{r}$ is a scalar since we are in one dimensional θ space although in higher dimensions it is typical to have different correlation lengths for each dimension. For larger values of r we see that the functions are very smooth and so the distribution at outputs for inputs further away from one another are more correlated than if we had a smaller value for r. Decreasing the value of this parameter results in realisations that are more 'wiggly' and so there is only non-negligible correlation between points that are very close to each other.

In Figure 4.2 we vary the scaling parameter σ^2. The effect of this is as expected, where we see that increasing the value of σ^2 results in larger error bands and uncertainty.

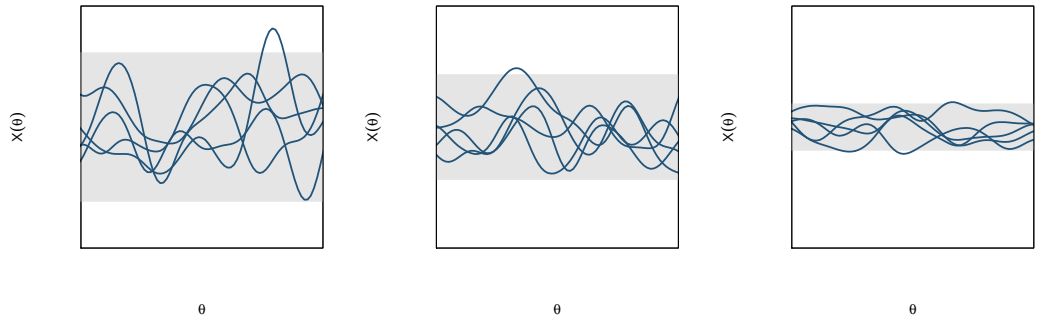

Figure 4.2: Random draws from a GP with squared exponential covariance function and hyperparameters $\sigma^2 = 1$ (left), $\sigma^2 = 0.5$ (middle) and $\sigma^2 = 0.1$ (right) with $r = 0.1$. Grey bands represent 95% probability intervals.

4.2.3 Prediction

Suppose that we have data $\mathcal{D} = \{(\boldsymbol{\theta}_i, x_i), i = 1, \ldots, N\}$ where $x_i = x(\boldsymbol{\theta}_i)$. Consider the task of inferring the function at a new input $\boldsymbol{\theta}^*$ for which the corresponding output is denoted X^*. Under a GP prior for $x(\boldsymbol{\theta})$ we have that

$$X^*(\boldsymbol{\theta}^*) \equiv X^* \sim N(m(\boldsymbol{\theta}^*), K(\boldsymbol{\theta}^*, \boldsymbol{\theta}^*)).$$

Similarly, $\boldsymbol{X} \sim N(m(\boldsymbol{\Theta}_N), K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N))$ using (4.1). Now,

$$\begin{aligned} Cov(X^*, \boldsymbol{X}) &= Cov(X^*(\boldsymbol{\theta}^*), X(\boldsymbol{\Theta}_N)) \\ &= K(\boldsymbol{\theta}^*, \boldsymbol{\Theta}_N) \end{aligned}$$

where $K(\boldsymbol{\theta}^*, \boldsymbol{\Theta}_N)$ is defined analogously to (4.2). Similarly, $Cov(\boldsymbol{X}, X^*) = K(\boldsymbol{\Theta}_N, \boldsymbol{\theta}^*)$. Hence, the joint distribution of $\boldsymbol{X}$ and X^* is given by

$$\begin{pmatrix} \boldsymbol{X} \\ X^* \end{pmatrix} \sim N\left\{ \begin{pmatrix} m(\boldsymbol{\Theta}_N) \\ m(\boldsymbol{\theta}^*) \end{pmatrix}, \begin{pmatrix} K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N) & K(\boldsymbol{\Theta}_N, \boldsymbol{\theta}^*) \\ K(\boldsymbol{\theta}^*, \boldsymbol{\Theta}_N) & K(\boldsymbol{\theta}^*, \boldsymbol{\theta}^*) \end{pmatrix} \right\}.$$

Conditioning on the observations $\boldsymbol{x} = (x_1, \ldots, x_N)^\top$ using multivariate normal results (A.2) gives

$$X^* | \boldsymbol{X} = \boldsymbol{x} \sim N(\boldsymbol{m}^*, K^*), \tag{4.5}$$

where

$$\boldsymbol{m}^*(\boldsymbol{\theta}^*) = m(\boldsymbol{\theta}^*) + K(\boldsymbol{\theta}^*, \boldsymbol{\Theta}_N) K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)^{-1} \{\boldsymbol{x} - m(\boldsymbol{\Theta}_N)\} \tag{4.6}$$

and

$$K^*(\boldsymbol{\theta}^*) = K(\boldsymbol{\theta}^*, \boldsymbol{\theta}^*) - K(\boldsymbol{\theta}^*, \boldsymbol{\Theta}_N) K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)^{-1} K(\boldsymbol{\Theta}_N, \boldsymbol{\theta}^*). \tag{4.7}$$

Similarly, if we wish to infer the function at a collection of inputs $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_{N^*}^*)^\top$ with corresponding outputs $\boldsymbol{X}^* = (X(\boldsymbol{\theta}_1^*), \ldots, X(\boldsymbol{\theta}_{N^*}^*))^\top$, we have that

$$\boldsymbol{X}^* | \boldsymbol{X} = \boldsymbol{x} \sim N(\boldsymbol{m}^*, K^*)$$

where $\boldsymbol{m}^*$ and K^* become

$$\boldsymbol{m}^*(\boldsymbol{\Theta}^*) = m(\boldsymbol{\Theta}^*) + K(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}_N) K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)^{-1} \{\boldsymbol{x} - m(\boldsymbol{\Theta}_N)\}$$

and

$$K^*(\boldsymbol{\Theta}^*) = K(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}^*) - K(\boldsymbol{\Theta}^*, \boldsymbol{\Theta}_N) K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N)^{-1} K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}^*).$$

Figure 4.3 gives an illustrative example of how we can use GP regression. In the first plot we see random function draws from the GP prior where we have a constant mean function and squared exponential covariance kernel. Once we observe some data in the second plot, we can use the above methods to find the fitted GP shown in the third plot, where we see the mean function adjusts to the training data and the 95% probability bands also tighten as we move closer to observations. In the space between the training data, the fitted mean will revert back to the prior mean function and the error bands will also increase due to the lack of data and increased uncertainty. This is particularly noticeable at the two end points of the plot. The final plot shows how the GP error bands become even tighter if we observe more data.

4.2.4 Inferring the hyperparameters

The mean and covariance functions of the GP depend on hyperparameters $\boldsymbol{\beta}, \sigma^2$ and $\boldsymbol{r}$ which need to be inferred from the (training) data. A question that arises here is how best to choose these hyperparameters. We can proceed by inferring the hyperparameters of GPs through Bayesian techniques as discussed in Chapter 3. We first specify a prior distribution on the hyperparameters, typically via the independent prior specification for each hyperparameter block

$$\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r}) = \pi(\boldsymbol{\beta}) \pi(\sigma^2) \pi(\boldsymbol{r}).$$

The posterior is then given up to proportionality as

$$\pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r} | \boldsymbol{x}) \propto \pi(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r}) L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r} | \boldsymbol{x}) \tag{4.8}$$

where $L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r} | \boldsymbol{x})$ is the likelihood under the GP. We shall omit the dependence on the hyperparameters in the covariance matrix and mean vector and write $K(\boldsymbol{\Theta}_N, \boldsymbol{\Theta}_N | \sigma^2, \boldsymbol{r}) =$

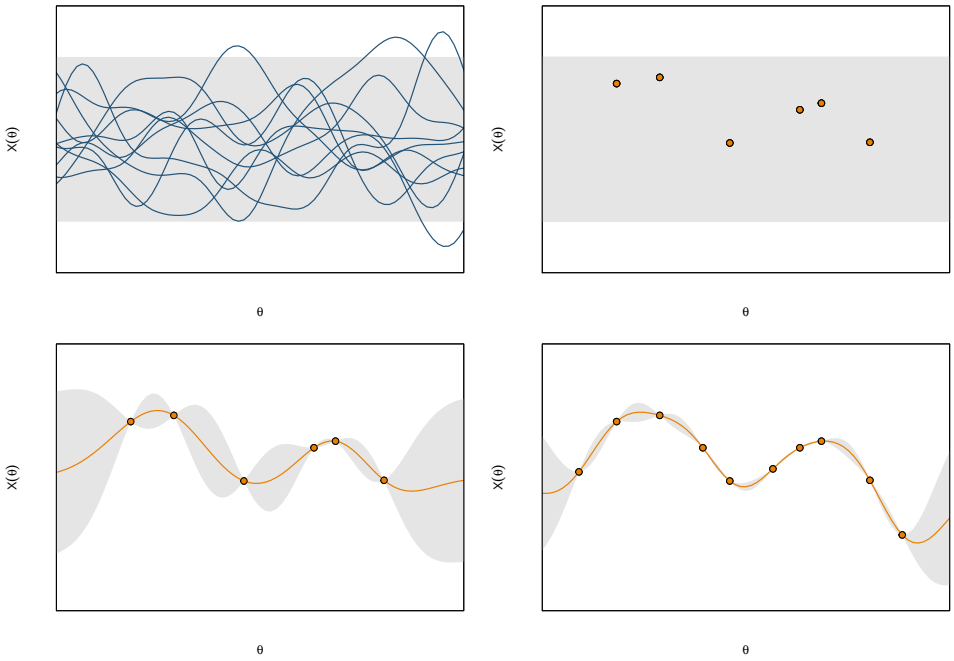

Figure 4.3: Random draws from a GP (top left), training data (top right), fitted GP (bottom left) and fitted GP with more observations (bottom right). Grey bands represent 95% probability intervals.

K for notational simplicity. Hence we obtain

$$L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r}|\boldsymbol{x}) = (2\pi)^{-N/2}|K|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N))^\top K^{-1}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N))\right), \quad (4.9)$$

and the log-likelihood is

$$l(\boldsymbol{\beta}, \sigma^2, \boldsymbol{r}|\boldsymbol{x}) = \log L = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|K| - \frac{1}{2}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N))^\top K^{-1}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N)). \quad (4.10)$$

As discussed by Rasmussen and Williams (2005), each of the terms of the log-likelihood in equation (4.10) have the following interpretations: the only term involving the data is the data-fit $\frac{1}{2}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N))^\top K^{-1}(\boldsymbol{x} - m(\boldsymbol{\Theta}_N))$; $\frac{1}{2}\log|K|$ is the complexity penalty depending only on the covariance function and the inputs and $\frac{N}{2}\log 2\pi$ is a normalization constant.

Naturally, the full posterior in (4.8) is intractable, given the complex dependence of the likelihood on σ^2 and $\boldsymbol{r}$. It should be noted that if the mean function is taken to be (4.3), a semi-conjugate prior specification is possible for $\boldsymbol{\beta}$, allowing a Gibbsian update of this hyperparameter, by placing a normal inverse gamma prior on $(\boldsymbol{\beta}, \sigma^2)^T$. Another simple approach adopted by Baggaley et al. (2012), among others, is to fix $\boldsymbol{\beta}$ at estimates obtained from a simple least squares fit.

We can appeal to MCMC methods as discussed in Section 3.3 to draw samples from

the posterior distributions for these hyperparameters. Alternatively, as discussed in Rasmussen and Williams (2005), we could optimise the log-likelihood in (4.10) over the hyperparameters in order to set the hyperparameters of the GP prior, a method known as empirical Bayes.

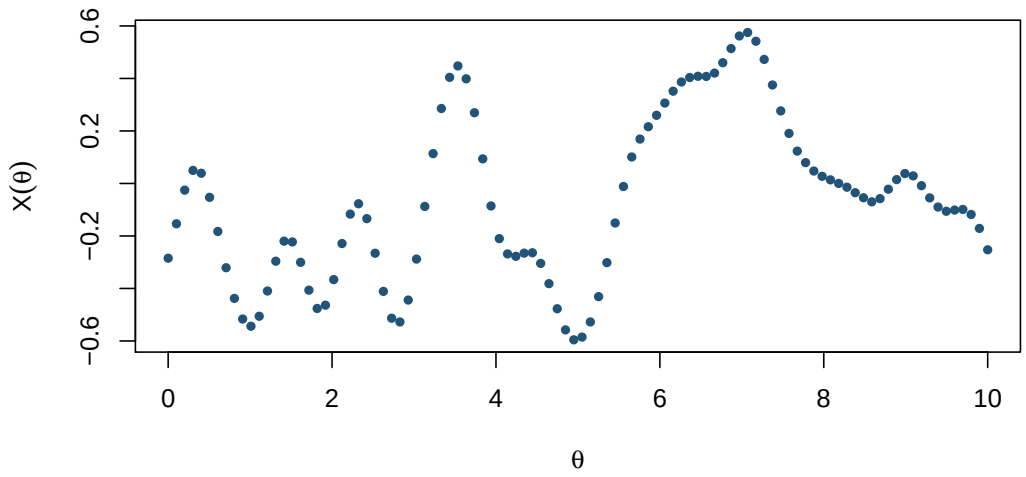

Figure 4.4: Simulated data from a GP.

4.2.5 Example: Simulation study

In this section we will study a synthetic dataset generated from a Gaussian process and infer the hyperparameters using MCMC to demonstrate how we can learn the hyperparameters of the covariance function. We took

$$X(\theta) \sim GP(\mathbf{0}, k(\theta, \theta'))$$

with covariance function of the form

$$k(\theta, \theta') = \sigma^2 \exp\left\{\frac{(\theta - \theta')^2}{r^2}\right\}.$$

Figure 4.4 shows the simulated data generated from the Gaussian process. We assume a zero mean function and infer the scale σ^2 and correlation length r, where the true values used to draw the data were $\sigma^2 = 0.1$ and $r = 0.5$ with $N = 100$. Since we are performing Bayesian inference we require prior distributions on each of these hyperparameters. We chose distributions that represent fairly vague prior beliefs about the parameters as shown

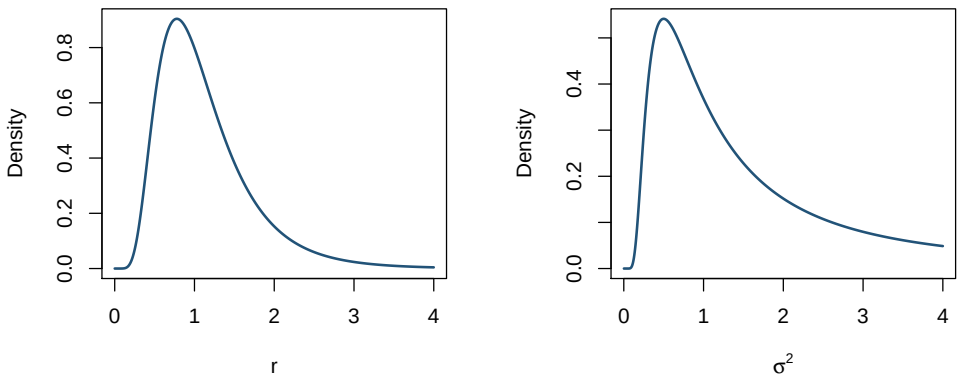

Figure 4.5: Left: Prior density for r. Right: Prior density for σ^2.

in Figure 4.5. We choose a lognormal prior for r and inverse gamma prior for σ^2,

$$r \sim LN(0, 0.5)$$

and

$$\sigma^2 \sim IG(1, 1).$$

Since it is not possible to specify a conjugate prior here, we used Metropolis-Hastings updates for each of the parameters which were treated as separate blocks. Since both parameters are positive, we use lognormal random walks as our proposal mechanism. That is, at iteration j we propose

$$\psi_1^* = \psi_1^{(j)} + w_1^{(j)}, \quad \psi_2^* = \psi_2^{(j)} + w_2^{(j)},$$

where $w_i^{(j)} \overset{indep}{\sim} N(0, \tau_i^2), i = 1, 2$ and $\boldsymbol{\psi} = (\log r, \log \sigma^2)^\top$. The tuning parameters τ_i^2 were chosen following advice in Section 3.3.2.

Performing an initial run of 1K iterations for tuning and then performing a longer run of 20K, we keep every 10^{th} iterate to obtain the posteriors shown in Figure 4.6. The 'true' values used to simulate the data are found in orange on the density plots, where we can see that the marginal hyperparameter posteriors are consistent with the true values that generated the data.

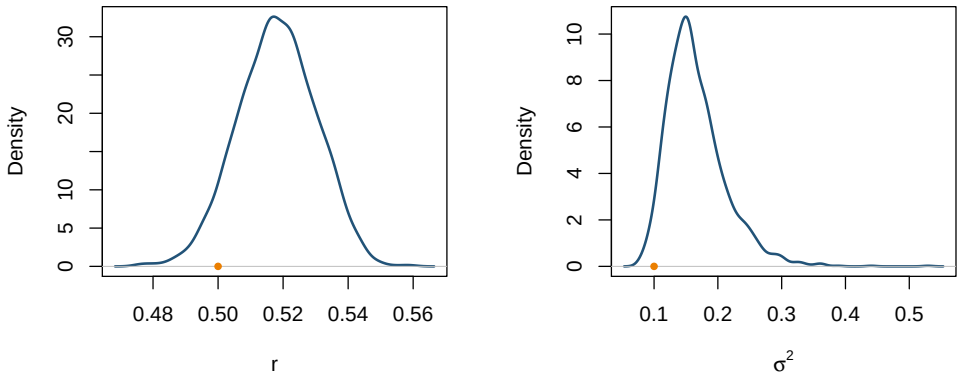

Figure 4.6: Left: Posterior density for r. Right: Posterior density for σ^2.

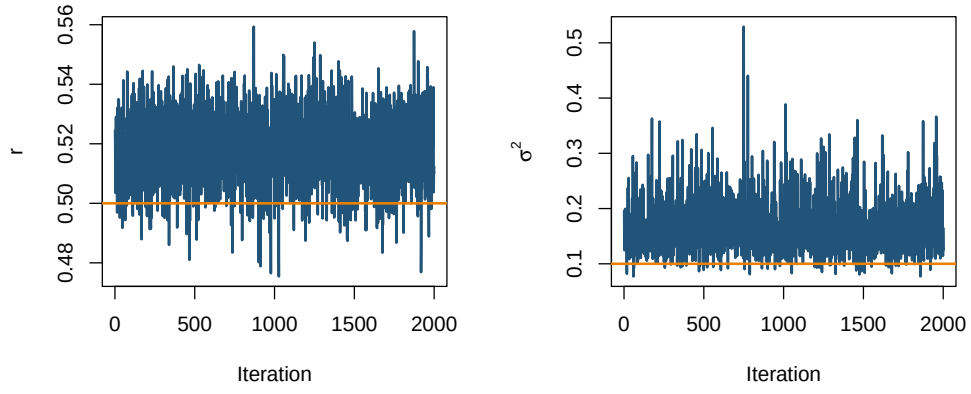

Figure 4.7: Left: Trace plot of samples for r. Right: Trace plot of samples for σ^2.

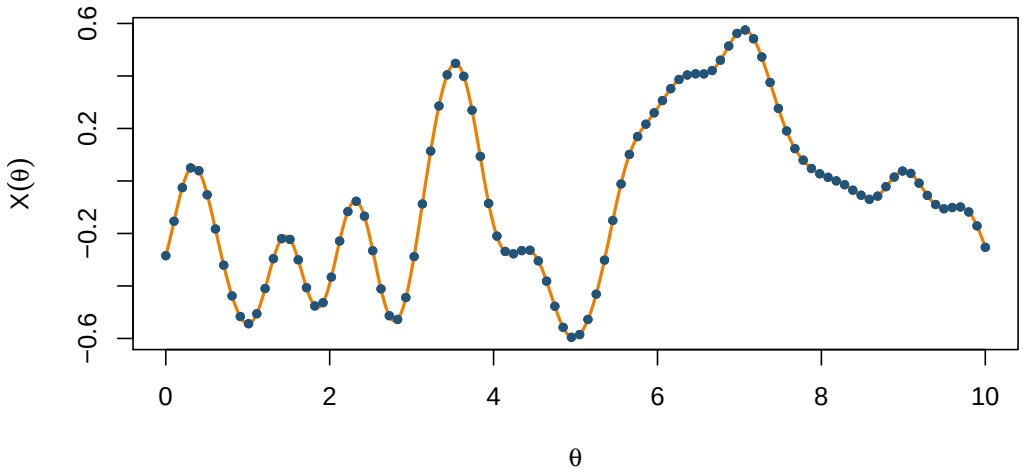

Figure 4.8: Fitted GP with grey 95% bands.

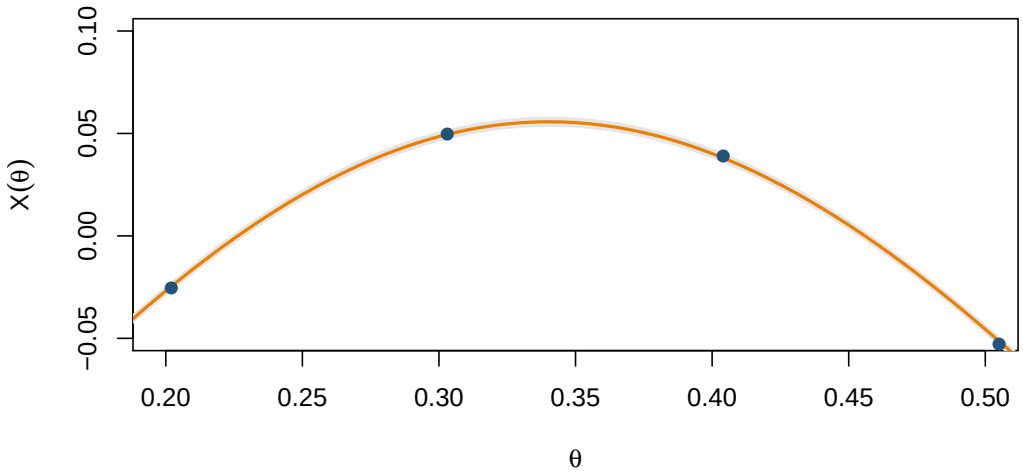

Figure 4.9: Fitted GP with grey 95% bands (close-up).

Chapter 5

Emulation

5.1 Introduction

In this chapter we consider the task of constructing an appropriate emulator. The use of Gaussian processes (GPs) to construct emulators is described in Sacks et al. (1989), Currin et al. (1991) and Kennedy and O'Hagan (2001). The idea is to obtain training data by running the expensive simulator a limited number of times and fit a GP to the resulting output.

Using a GP in this way allows for fully probabilistic interpolation between output values for which there is no input in the training set.

Emulators are commonly used for approximating deterministic processes where repeat simulations at the same input gives the same output. This is not the case for stochastic kinetic models, which will produce different outputs for the same input, in this case the rate constants. Moreover, for the Schlögl system, there is more than one stable state. Recent work has looked at using emulators for stochastic output (Henderson et al., 2009) within the context of SKMs where a GP is used to model the mean and variance, which are smooth nonlinear deterministic functions of the inputs.

We shall consider the task of using a single GP to emulate the prey output of the Lotka-Volterra model as well as consider some common diagnostic tools.

5.2 Training data design

In order to use a GP for emulation, we must first *train* the GP with training data. In the context of emulating SKM output, the training data will be simulator output of the SKM obtained via Gillespie's direct method, as discussed in Section 2.4.1. For a given number

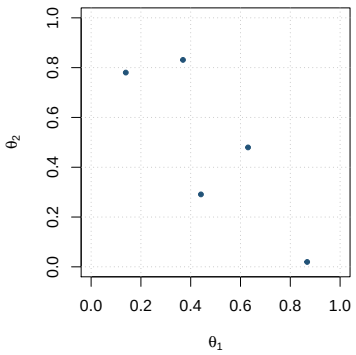

Figure 5.1: Latin hypercube design in two-dimensions with 5 design points.

of training points (which may be dictated by computational budget), we wish to maximise coverage of the input space. A poor choice of design can lead to parts of the design space not being well explored which in turn can lead to an inaccurate emulator in certain parts of the input space. We therefore aim to acquire points that are well spread out over the design space; a *space filling* design.

One possible approach is to use a Latin hypercube design by utilising Latin hypercube sampling which was first described by McKay et al. (1979). A Latin hypercube is a generalisation of the two-dimesional equivalent Latin square. A Latin square is defined as a square grid where there is only one point in each row and each column as shown in Figure 5.1. When the Latin hypercube is projected onto either axis, the points provide good coverage in that dimension. This generalises to higher dimensions, where projecting the points onto any subset of the input axes produces a well covered design. However, it should be noted that this approach does not necessarily ensure the whole space is well covered. In the most extreme case, for two dimensions, the points could lie precisely on the diagonal and still produce a valid Latin square. To tackle this potential issue, Morris and Mitchell (1995) proposed the maximin Latin hypercube design which works by maximising the distance between points in a Latin hypercube sample. This is achieved by producing a list of the minimum distance between points for each design using the Euclidean distance $d(\boldsymbol{\theta}, \boldsymbol{\theta}')$. The maximin design maximises the minimum distance between design points. A comparison of the regular Latin hypercube design and the maximin design can be found in Figure 5.2 which demonstrates how the maximin design produces a more even spread of the design space providing a favourable set of points over the regular Latin hypercube design. This was achieved by making use of the Latin hypercube samples package (Carnell, 2016) in R (R Core Team, 2016).

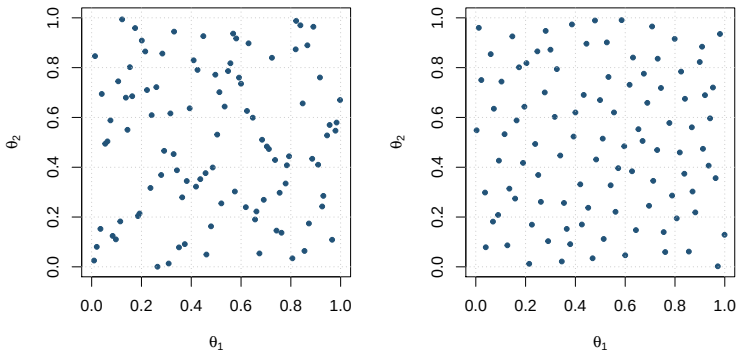

Figure 5.2: Two dimensional designs with 100 points. Left: Latin hypercube design. Right: Maximin Latin hypercube design.

5.2.1 Application to SKMs

Suppose that we have observations on

$$Y(t) = F^\top X(t) + \epsilon_t, \qquad \epsilon_t \overset{indep}{\sim} N(0, \Sigma), t = 1, \ldots, T \tag{5.1}$$

where F is an n (number of species) $\times$ d (number of observed components) constant matrix allowing for observation of a subset of components of $X(t)$. Recall from Section 3.4.1 that the joint posterior is given by

$$\pi(\boldsymbol{\theta}, \Sigma, \boldsymbol{x}|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\pi(\Sigma)\pi(\boldsymbol{x}|\boldsymbol{\theta})\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma)$$

where in the case of (5.1)

$$\pi(\boldsymbol{y}|\boldsymbol{x}, \Sigma) = \prod_{t=1}^{T} N(\boldsymbol{y}_t; F^\top \boldsymbol{x}_t, \Sigma)$$

and $N(\cdot; \boldsymbol{m}, V)$ denotes the density of a multivariate Gaussian random variable with mean vector $\boldsymbol{m}$ and variance matrix V.

Given observed data $\boldsymbol{y}$ and the training data, it is possible in theory to fit the emulator and statistical model (5.1) jointly using an MCMC scheme. However, this is likely to be extremely computationally expensive compared to fitting the emulator and statistical model separately. By fitting separately, we can fit each emulator at each time point independently and in parallel and then fit the statistical model. This approach is advocated by Bayarri et al. (2007) and adopted by Henderson et al. (2009) and Baggaley et al. (2012) among others.

Rather than build a complex emulator whose inputs are both the log-rate parameters and time, we propose to pragmatically build parameter only emulators at each observed time-point. This has the advantage that the emulators can be built and evaluated/sampled in parallel.

Recall the likelihood-free Metropolis-Hastings scheme of Algorithm 4. A likelihood-free Metropolis-Hastings scheme that uses the (fitted) GP surrogate as the inferential model can be implemented straightforwardly by replacing draws from the MJP simulator in step 3, with draws from (4.5).

If interest lies primarily in inference for $\boldsymbol{\theta}$, a more efficient Metropolis-Hastings scheme can be constructed by noting that under the GP and (5.1) it is possible to integrate out $\boldsymbol{x}$. Using a subscript e to denote use of an emulator, we have the approximate posterior

$$\pi_e(\boldsymbol{\theta}, \Sigma|\boldsymbol{y}) \propto \pi(\boldsymbol{\theta})\pi(\Sigma)\pi_e(\boldsymbol{y}|\boldsymbol{\theta}, \Sigma) \tag{5.2}$$

where

$$\pi_e(\boldsymbol{y}|\boldsymbol{\theta}, \Sigma) \propto |V^*(\boldsymbol{\theta})|^{-1/2} \exp\left\{-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{M}^*(\boldsymbol{\theta}))^\top V^*(\boldsymbol{\theta})^{-1}(\boldsymbol{y} - \boldsymbol{M}^*(\boldsymbol{\theta}))\right\},$$

where $\boldsymbol{M}^*(\boldsymbol{\theta})$ has t^{th} element given by $\boldsymbol{m}_t^*(\boldsymbol{\theta})$ (4.6) and $V^*(\boldsymbol{\theta}) = \text{diag}\{K_t^*(\boldsymbol{\theta}) + \Sigma\}$ (4.7).

Finally, we note that the fitted GP, as described, would give exactly the training output when evaluated at the training input. Accounting for the uncertainty in the SKM output can be done in a number of ways e.g. by modifying the covariance function by including an additional term for the case $\boldsymbol{\theta}' = \boldsymbol{\theta}$ or by following a similar method to Henderson et al. (2010) where SKM output would be emulated with a Gaussian distribution where the mean and variance are both nonlinear deterministic functions derived from the (posterior) predictive means of two independent GPs. We shall consider the former of these two methods in what follows.

5.3 Example: Lotka-Volterra prey output

In this section we consider the use of a GP to emulate the prey output from the Lotka-Volterra model (Section 2.6) at a single point in time. The system is described by three

reactions

$$\begin{aligned} R_1 : \quad & X_1 & \xrightarrow{e^{\theta_1}} \quad & 2X_1, \\ R_2 : \quad & X_1 + X_2 & \xrightarrow{e^{\theta_2}} \quad & 2X_2, \\ R_3 : \quad & X_2 & \xrightarrow{e^{\theta_3}} \quad & \emptyset. \end{aligned}$$

where the rate constants $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^\top$ are the parameters we would want to infer in our calibration process given some noisy data.

To demonstrate how to use a GP for emulation of stochastic kinetic model output we will emulate the prey species (X_1) of this model. We will keep two of the three rate constants (θ_1, θ_2) fixed so that we can easily visualise the GP in one-dimensional θ_3 space.

We first need to train our emulator with training data which we generate using Gillespie's direct method at a set of design points in θ_3 space. Since this is a one-dimensional case we can simply make our set of N training points an evenly spaced grid of points between two bounds. If we had more than one-dimension we could use a Latin hypercube design (Section 5.2) to ensure our design points were optimally spaced. In a calibration setting we could, for example, choose the bounds to be in the tails of the prior distribution of the rate constants. For this illustrative example, we shall use values that are well within the tails of the marginal posterior for θ_3 as found in the analysis in Golightly and Wilkinson (2011). The bounds are chosen to be $\log 0.15 < \theta_3 < \log 0.3$ and we fix $\theta_1 = \log 0.5$ and $\theta_2 = \log 0.0025$. These are typical values chosen to give the oscillatory behaviour seen in Section 2.6.

When emulating deterministic output, repeat simulations at the same input will give the same output. This is not the case for stochastic output as repeat simulations will give similar but different results. For this reason we will require M replicate observations of the simulator output at each of our N design points in order to capture the variation in the output. The choice of both N and M represents a trade-off between speed and accuracy; choosing these to be too large results in a more accurate emulator but at greater computational cost. For this example we choose $N = 10$ design points with $M = 5$ replicates at each point, which should be sufficient in providing an accurate emulator. We use Gillespie's direct method to obtain simulator output at our design points therefore forming our training data (Figure 5.3). We fit the following GP to the training data where we drop subscripts for X_1 and θ_3 for ease of exposition,

$$X(\theta) \sim GP(\mathbf{0}, k)$$

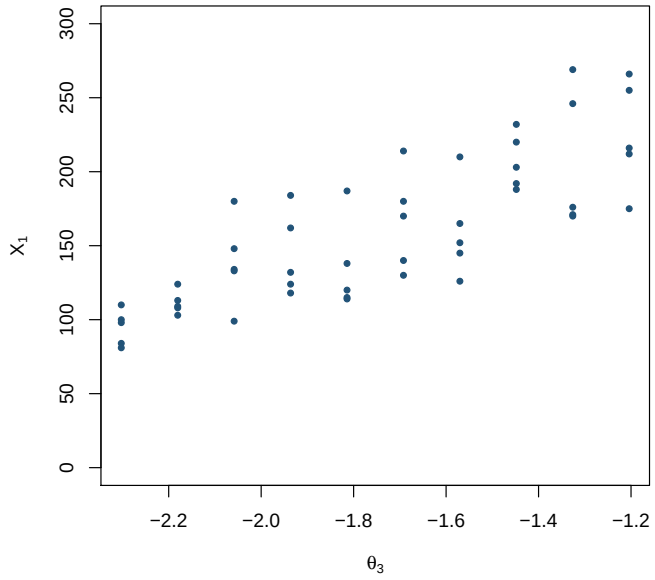

Figure 5.3: Training data at $N = 10$ points in θ_3 space.

with covariance function of the form

$$k(\theta_i, \theta_j) = \sigma^2 \exp\left\{\frac{(\theta_i - \theta_j)^2}{r^2}\right\} + \tau^2 \delta_{ij}.$$

Note we are modelling stochastic output so the extra term $\tau^2\delta_{ij}$ in the covariance function is necessary to account for uncertainty in the output for a given input. As described in Section 4.2.5, we appeal to MCMC methods to infer the hyperparameters of the GP where we use Metropolis-Hastings updates for each of the parameters. We place a lognormal prior on r and inverse gamma priors on σ^2 and τ^2,

$$r \sim LN(0, 0.5),\ \sigma^2 \sim IG(1, 1) \text{ and } \tau^2 \sim IG(1, 1).$$

Performing an initial run of 1K iterations for tuning and then performing a longer run of 20K, we keep every 10^{th} iterate to reduce autocorrelation. We obtain the marginal posteriors densities shown in Figure 5.4. By fixing the hyperparameters at their posterior means we can obtain the fitted GP shown in Figure 5.6 which demonstrates how well the GP works as an emulator in this scenario when we have stochastic output. Averaging over the posterior uncertainty in the hyperparameters is also possible although Henderson et al. (2009) find that fixing at the posterior means has a negligible effect on predictive

performance. In a calibration setting we would need to interpolate the distribution of stochastic kinetic model output at new points in θ-space which is now straightforward to do using our emulator. We would simply condition on the training data, as described in Section 4.2.3, allowing us to interpolate the distribution at new points in space. In the next section we look at a number of diagnositics to assess the emulator fit.

5.3.1 Diagnostics

After we have constructed our emulator we can assess how well it performs as an approximate replacement for our simulator. We discuss two diagnostics below and direct the reader to Bastos and OHagan (2009) for a detailed overview of diagnostics for emulators.

In the following examples of diagnostics we require a set of data which will be used for validation. If the input space was greater than one-dimension then we could use a new Latin hypercube design over the same input space and use our simulator at each point to give us our validation data. However, it may be infeasible to generate a new set of data to use for validation and so diagnostics must therefore be calculated using the existing training data. In this case we could appeal to a 'leave-one-out' approach (Rougier et al., 2009) where one data point is removed from the training data, the emulator is fit on the remaining training data and the omitted data point is used for diagnositics.

Since our Lotka-Volterra example above is one-dimensional we simply use a new grid of N^* points over the same input space. Because of the stochasticity in our outputs we also have M^* replicates at each of these points, just as we did in our training data. This new data $(\boldsymbol{\theta}^*, \boldsymbol{x}^*)$ will then be used for validation. For the above example we use a validation set of $N^* = 30$ with $M^* = 5$ replicates.

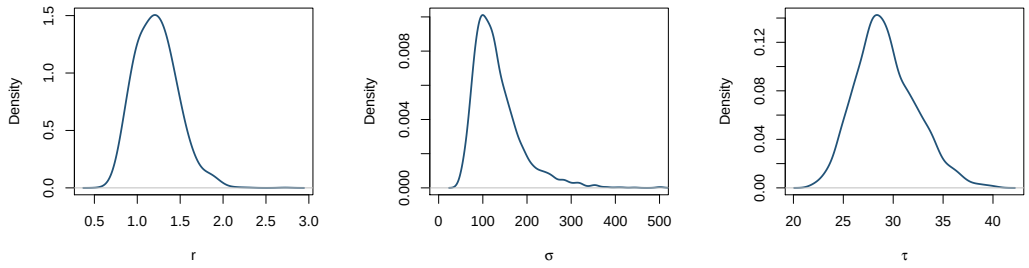

Figure 5.4: Left: Posterior density for r. Middle: Posterior density for σ. Right: Posterior density for τ.

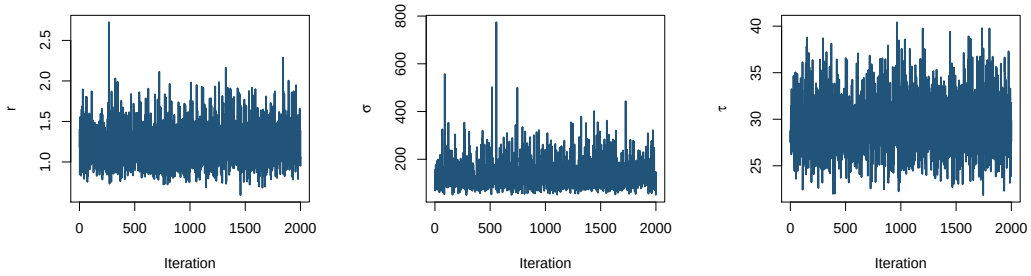

Figure 5.5: Left: Trace plot of samples for r. Middle: Trace plot of samples for σ. Right: Trace plot of samples for τ.

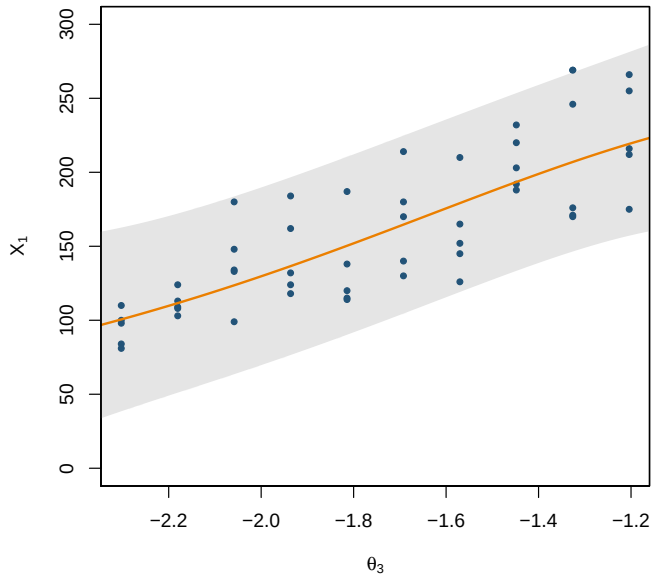

Figure 5.6: Fitted GP mean function with grey 95% bands.

Probability integral transform

As described in Gneiting et al. (2007), the probability integral transform (PIT) can be used to check that the Gaussian assumption is valid for our emulator. The PIT is defined as

$$D_{\text{PIT}}(\theta_i^*) = \Phi\left[\frac{x(\theta_i^*) - m^*(\theta_i^*)}{\sqrt{k^*(\theta_i^*, \theta_i^*)}}\right]$$

for each datum in our validation set. If the Gaussian assumption of our emulator is correct then $D_{\text{PIT}}(\theta_i^*)$ should have a standard uniform distribution. Plotting a histogram of these diagnostics (Figure 5.7, left) for the Lotka-Volterra example above we see that the assumptions seem valid as the histogram looks reasonably flat. Deviations from this would suggest that the distributional assumptions of the emulator are not valid.

Individual prediction errors

Another diagnostic we can use to assess how well our emulator is fitting is to calculate the individual prediction errors (Bastos and OHagan, 2009). The individual prediction errors are defined as

$$D_{\text{IPE}}(\theta_i^*) = \frac{x(\theta_i^*) - m^*(\theta_i^*)}{\sqrt{k^*(\theta_i^*, \theta_i^*)}},$$

for each datum in our validation set. If the emulator can suitably represent the simulator then these prediction errors should be approximately normal. We should therefore expect approximately 95% of these prediction errors to be within the interval $(-2, 2)$. When plotting the prediction errors we should expect a random scatter with no obvious patterns as this could suggest that the stationarity assumption is not appropriate. If we look at Figure 5.7 (right) we can see the majority of our errors are within the interval, suggesting the emulator is fitting adequately.

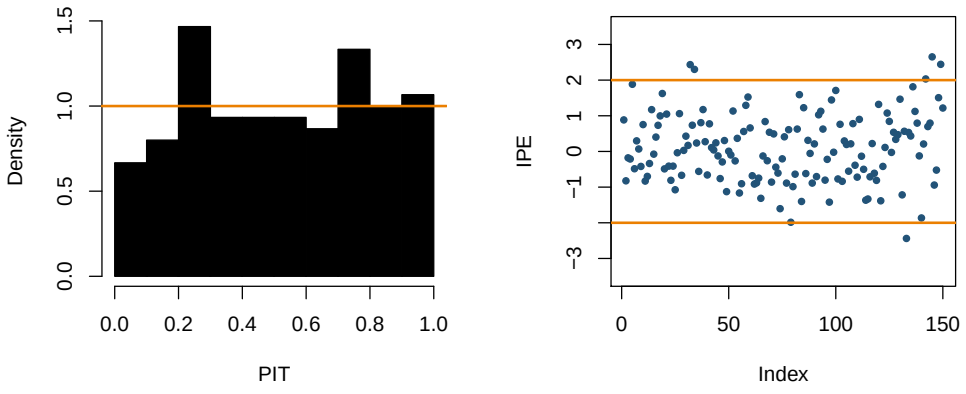

Figure 5.7: Diagnostics for the Lotka-Volterra emulator.

Chapter 6

Dirichlet processes

6.1 Introduction

Within the context of SKMs, a satisfactory emulator should account for output uncertainty at the same input and be able to account for multimodality, as exhibted, for example, by the Schlögl system (Section 2.7).

In Chapter 4 we defined Gaussian processes and in Chapter 5 we explained how they can be used to emulate output and interpolate between training points. We now consider an emulator based on a mixture of Gaussian processes. By using a mixture we will be able to model the bimodal nature that the Schlögl system exhibits. Of course we could use a two-component mixture of Gaussian processes and assume that this would adequately model the Schlögl output at certain times. We propose to avoid this ad hoc approach by placing a *Dirichlet process prior* on the number of components.

In this chapter we provide the necessary background on Dirichlet processes. We begin by introducing finite mixture models in the simple context of normal random variables. We then move on to the Dirichlet process and give a formal definition and several intuitive representations including the well-known Chinese restaurant process. We then demonstrate how the process can be used in mixture modelling (Ferguson (1973), Antoniak (1974)) and look at two examples where we have a simple mixture of normals, and also a two-component regression mixture.

6.2 Finite mixture modelling

Mixture modelling allows us to describe a potentially complex distribution using a collection of much simpler distributions. Instead of assuming that all of the data was drawn

from a single distribution, the data could be made up of multiple sub-populations. For this reason, mixture models are widely used in density estimation and clustering problems.

We begin by considering univariate data for which the general form of a finite mixture model is

$$f(x|\boldsymbol{\theta}) = \sum_{j=1}^{k} \pi_j f_j(x|\theta_j),$$

where k is the number of groups, or *components*, and θ_j are the component-specific parameters of the j^{th} component and f_j is a probability density function or probability mass function depending on whether or not $\boldsymbol{x}$ is continuous or discrete. The parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ are the weights of each component such that

$$\sum_{j=1}^{k} \pi_j = 1, \qquad \pi_j > 0 \quad \forall j.$$

A popular choice of density is the normal density, in which case we obtain a Gaussian mixture model, where each component is parameterised by a mean and variance giving

$$f(x|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{j=1}^{k} \pi_j N(x; \mu_j, \sigma_j^2).$$

To demonstrate this approach using a very simple example, suppose we have some exchangeable scalar data $x_1, \ldots, x_n$ that we assume to be drawn from some unknown distribution shown in Figure 6.1. Fitting a two component Gaussian mixture model to these data will result in a bimodal distribution, where not only the means are different between the components but the variances also differ. Fitting a single normal distribution to this data may not be appropriate. We may also fit a three component Gaussian mixture which may provide a better fit to the data, but is less parsimonious than using fewer parameters. A potential problem with finite mixture modelling is specifying the number of components, a problem which can be alleviated by appealing to infinite mixture models. By their nature, infinite mixture models induce an infinite dimensional parameter space and thus fall within the area of Bayesian non-parametrics (Hjort et al., 2010).

6.3 Typical representations

The Dirichlet process is a conjugate prior for infinite dimensional categorical distributions – a generalisation (to infinite dimension) of the result that the Dirichlet distribution is a conjugate prior for the categorical (multinomial) distribution. We now provide an overview and describe the common representations of the Dirichlet process before considering in-

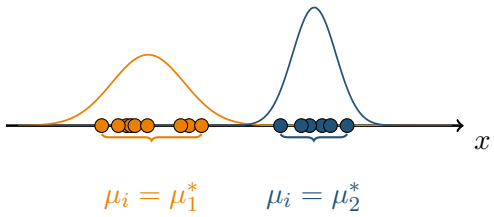

Figure 6.1: Gaussian mixture model

finite mixture models. The overview provided here is somewhat brief and we refer the reader to either Ferguson (1973) and Antoniak (1974) or the more recent book by Hjort et al. (2010) for further details on the underlying measure theory for Dirichlet processes.

Dirichlet processes were first introduced by Ferguson (1973) as a prior distribution for nonparametric analysis. Let G_0 be a distribution over some space Θ and $\alpha > 0$. Then, a distribution G is said to follow a Dirichlet process if

$$\big(G(A_1), \ldots, G(A_k)\big) \sim Dir\{\alpha G_0(A_1), \ldots, \alpha G_0(A_k)\} \tag{6.1}$$

for every finite measurable partition $A_1, \ldots, A_k$ of Θ.

The Dirichlet process is essentially an infinite generalisation of the Dirichlet distribution, somewhat analogous to how the Gaussian process is an infinite generalisation of the normal distribution. From the definition we see that all marginals of G are Dirichlet distributed. The distribution G_0 is called the base distribution and α is the concentration parameter. The base distribution is the expected value of the process, that is $\mathrm{E}(G) = G_0$, and must have the same support as G.

The concentration parameter controls how similar the process is to the base distribution. For $\alpha \to 0$, realisations of θ (the parameters from G where G follows a Dirichlet process) are concentrated around a single point and in the limit $\alpha \to \infty$ realisations become similar to the base. It follows that the probability of two distinct components (of realisations from G) being equal, $\Pr(\theta_i = \theta_j)$ for $i \neq j$, tends to 0 as $\alpha \to \infty$ (assuming G_0 is continuous) whereas $\Pr(\theta_i = \theta_j) \to 1$ as $\alpha \to 0$. This is illustrated in Figure 6.2 where each row has three independent realisations from a Dirichlet process with $\alpha = 1, 10, 100$ from top to bottom respectively. The base distribution in this case is chosen to be $G_0 = N(0, 1)$. We observe that for the smaller values of α our (discrete distribution) realisations G are defined over fewer atoms with some having large mass. It follows that (theoretically) if $\alpha = 0$ we would have a single point mass at some value $\theta \in \mathbb{R}$. For larger values of α we see that the realisations (distributions) have increasingly more unique atoms, each having relatively small weight. If, in theory, $\alpha = \infty$ then this distribution would be defined over infinitely many atoms each of which has mass 0, that is, the distribution would be $G_0 = N(0, 1)$.

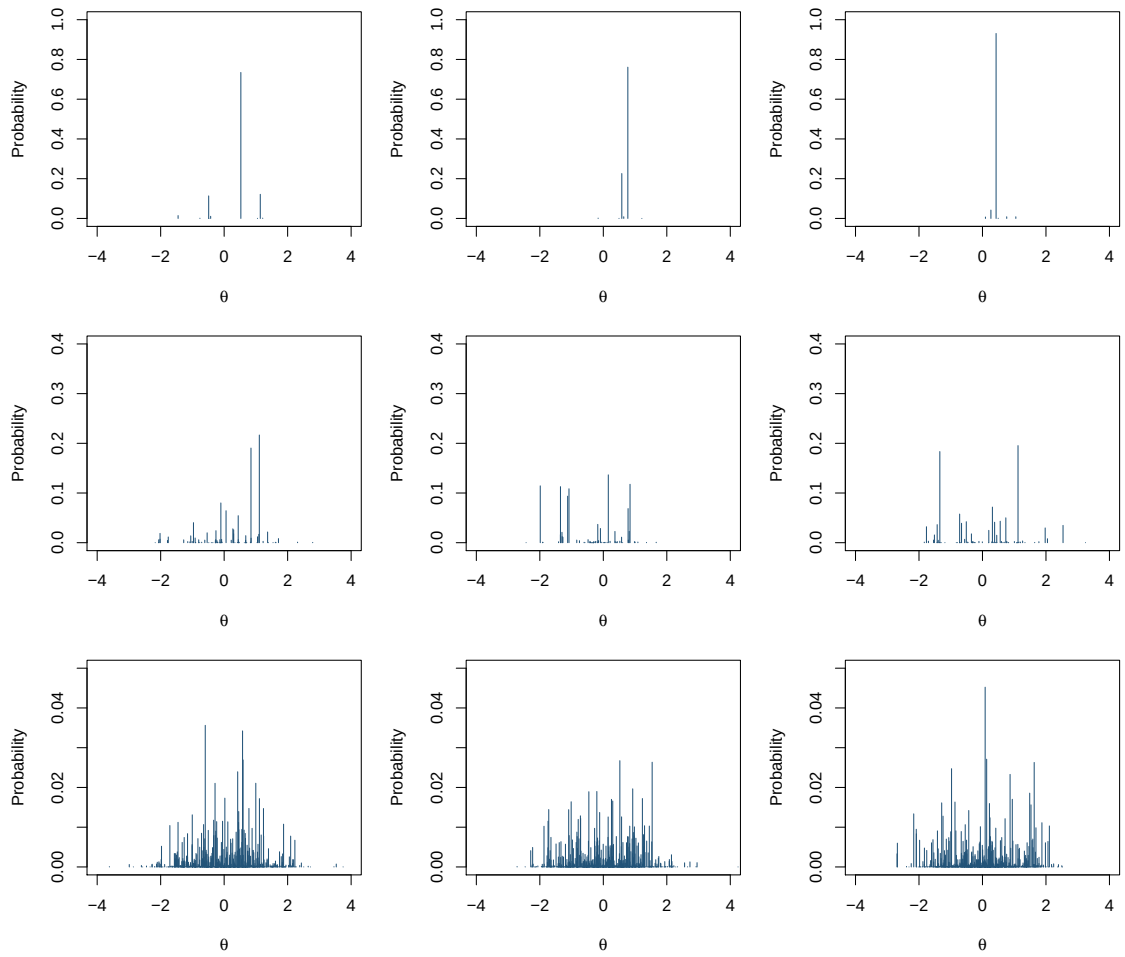

Figure 6.2: Multiple realisations from a Dirichlet process with $G_0 = N(0, 1)$ and $\alpha = 1, 10, 100$ from top to bottom respectively.

Figure 6.3 shows the convergence of G to the base distribution as α increases. This shows the empirical cumulative distribution function (CDF) for each of the respective realisations shown within Figure 6.2. As α increases, the CDF of these realisations becomes more like that of a $N(0, 1)$ distribution, that is, $G \to G_0$ as $\alpha \to \infty$.

There are a number of representations of the Dirichlet process, and we provide details of some commonly used ones here.

6.3.1 Stick-breaking process

The Dirichlet process can be represented as a stick-breaking process (Sethuraman, 1994).

Since the distribution drawn from a Dirichlet process is discrete, we can write the density

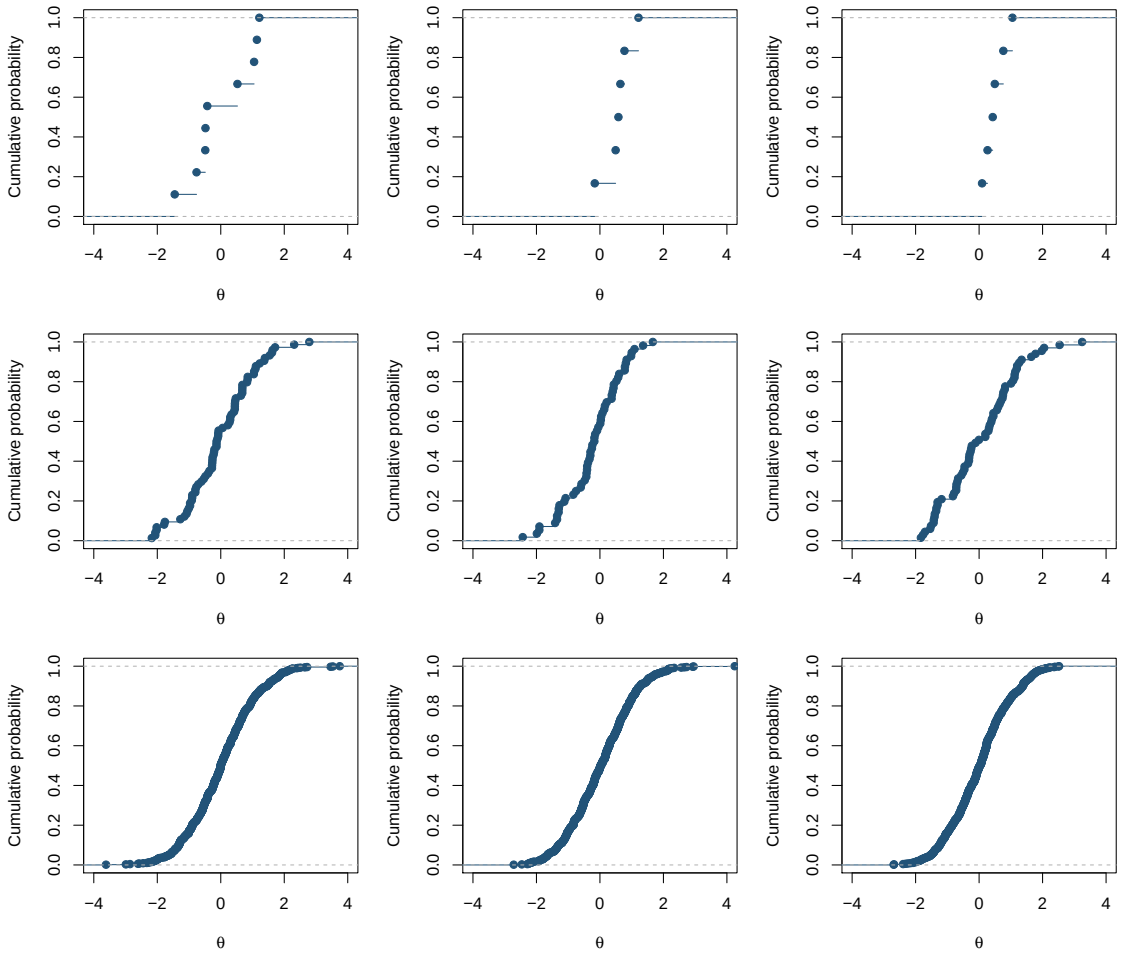

Figure 6.3: Empirical CDF from multiple realisations from a Dirichlet process with $G_0 = N(0, 1)$ and $\alpha = 1, 10, 100$ from top to bottom respectively.

function as

$$f(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta)$$

where $\{\theta_k\}_{k=1}^{\infty}$ are our parameters drawn from the base distribution (which we call *atoms*), $\{\pi_k\}_{k=1}^{\infty}$ are the corresponding weights and δ_{θ_k} is a Dirac mass function that equals 1 at θ_k and 0 elsewhere.

The parameters/atoms θ_k are each drawn from the base distribution of the Dirichlet process, G_0. The weights (or probabilities) are then constructed through a stick-breaking approach where we imagine a stick of unit length and we repeatedly break off segments of length π_k (Figure 6.4). This is achieved via beta random variables, where

$$\pi_k = \tilde{\pi}_k \prod_{l=1}^{k-1} (1 - \tilde{\pi}_l) \qquad \text{and} \qquad \tilde{\pi}_k \sim Beta(1, \alpha), \qquad k = 1, \ldots, \infty,$$

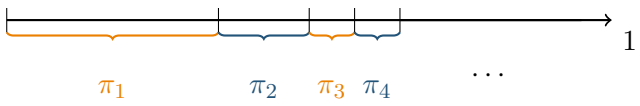

Figure 6.4: Stick-breaking illustration

and $\pi_1 = \tilde{\pi}_1$.

The role of α is easier to see in this representation. For instance, for small α the first component will have a large weight to it since a lot of the stick will be broken off. In contrast, large α will result in small lengths of the stick breaking off for each component.

Approximations can then be made through truncation (Ishwaran and Zarepour, 2002). The truncation parameter $N < \infty$ should be chosen such that the the first $N - 1$ stick lengths have accounted for most of the unit length stick and the remaining length is negligible. Setting $\tilde{\pi}_N = 1$ ensures that the weights sum to 1. The approximation can then be used as a Dirichlet process prior; see Ishwaran and James (2001) where the authors present Gibbs sampling methods for fitting models with stick-breaking priors thus allowing for the posterior to be approximated through MCMC.

6.3.2 Chinese restaurant process

Another representation of the Dirichlet process is through the Chinese restaurant process (Aldous, 1985).

We begin by imagining a Chinese restaurant (or any cuisine you like) with infinitely many tables. When a new customer enters the restaurant, they sit down at a table with probability proportional to the number of diners already at that table. With probability proportional to α the customer will sit at a new table by themselves. This process continues until all n customers have been seated. At this point $N^c \leq n$ tables will be occupied, and the individuals at each table are interpreted as being clustered together. It follows that N^c is the number of unique clusters. The phrase *'the rich get richer'* springs to mind here as tables with large numbers of people will have high probability of being the table to be chosen by the next customer.

This representation only specifies the distribution over the partitions, essentially only performing the clustering of the observations (or customers). Drawing independent realisations from G_0 and assigning a value to each table results in a discrete distribution with probabilities proportional to the number of customers at that table. This discrete distribution is a draw G which is a draw from a Dirichlet process with base distribution G_0 and concentration parameter α. This process is exchangeable, that is to say, the order in which the n customers arrive does not affect the final probability distribution.

6.3.3 Pólya urn scheme

Another way to visualize a Dirichlet process (and the associated Chinese restaurant process) is via a Pólya urn scheme (Blackwell and MacQueen, 1973). For this analogy it is useful to consider $\alpha \in \mathbb{N}$ although the probabilities of observations being assigned to each cluster (to be defined) hold for any $\alpha \in \mathbb{R}_{>0}$.

In this visualisation, we imagine we have an urn filled with α black balls. We then carry out the following steps. We draw a ball from the urn.

- If the ball is black, we return the ball back to the urn along with a *uniquely* coloured ball we have generated.
- If the ball drawn is *not* black we generate an additional ball of the same colour as the one drawn and return both back into the urn.

This process continues until n balls have been drawn. Once the nth ball has been drawn (and the appropriate action taken) the black balls are discarded from the urn. At this point there will be $N^c \leq n$ uniquely coloured balls within the urn and the distribution over these colours is equivalent to the distribution over the tables within the Chinese restaurant process. Drawing independent realisations from G_0 and assigning a value to each unique colour results in a discrete distribution with probabilities proportional to the number of each coloured ball. Again this process is also exchangeable, that is, if n people each select one ball, the order in which the people are arranged does not affect the final probability distribution.

The probability of assigning the ith person/ball to table/ball colour j under both of these alternative representations is

$$\begin{aligned} \Pr(c_i = j | c_1, \ldots, c_{i-1}) &= \frac{n_{ij}^c}{\alpha + i - 1}, \qquad \text{for } j = 1, \ldots, N^c, \\ \Pr(c_i = N^c + 1 | c_1, \ldots, c_{i-1}) &= \frac{\alpha}{\alpha + i - 1}, \end{aligned}$$

where n_{ij}^c denotes the current number of observations assigned to cluster j (at iteration i), c_i is the cluster indicator variable for observation (person/ball) i and N^c is the number of unique clusters that currently exist ($N^c = 0$ when $i = 1$).

6.3.4 Posterior distribution

Let $(\theta_1, \ldots, \theta_n)$ be a sequence of i.i.d.(independently and identically distributed) samples from G. Since G is a distribution over Θ, each $\theta_i \in \Theta$ for $i = 1, \ldots, n$. As above, $A_1, \ldots, A_k$ is a finite measurable partition of Θ and we let $n_j = \{i : \theta_i \in A_j\}$, that is, n_j is the number

of θ_i (observed samples) that are in the partition A_j. Using the conjugacy between the multinomial and Dirichlet distributions, we can find the posterior distribution of G as

$$\big(G(A_1), \ldots, G(A_k)|\theta_1, \ldots, \theta_n\big) \sim Dir\{\alpha G_0(A_1) + n_1, \ldots, \alpha G_0(A_k) + n_k\}.$$

Since this is true for all measurable paritions of Θ we recognise that the posterior distribution is also a Dirichlet process (Teh, 2011), and we write

$$G|\theta_1, \ldots, \theta_n \sim DP\big(\frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n}, \alpha + n\big).$$

6.3.5 Predictive distribution

We can use the same ideas as above, from Teh (2011), to find the predictive distribution $\theta_{n+1}|\theta_1, \ldots, \theta_n$, that is, upon marginalising out G. Given our i.i.d. samples $(\theta_1, \ldots, \theta_n)$, the predictive distribution for θ_{n+1} will be

$$\theta_{n+1}|\theta_1, \ldots, \theta_n \sim \frac{1}{\alpha + n}\big(\alpha G_0 + \sum_{i=1}^{n} \delta_{\theta_i}\big).$$

This is a mixture distribution where θ_{n+1} will be drawn from either the base distribution (thus a new cluster being created) or will be drawn from the empirical distribution $\sum_{i=1}^{n} \delta_{\theta_i}$ (thus the $n + 1^{th}$ observation joining an existing cluster). This is essentially what occurs when considering the Pólya urn representation (Blackwell and MacQueen, 1973) of the Dirichlet process. The colour of the next ball, θ_{n+1}, will depend on the colours of all of the balls drawn previously $(\theta_1, \ldots, \theta_n)$.

6.4 Dirichlet Process Mixture modelling

One problem with mixture modelling is determining the number of components to fit *a priori*. Use of a Dirichlet process (DP) in mixture modelling allows us to avoid specifying the number of components, as this essentially induces a countably infinite number of components, but only certain components will have data assigned to them.

As a simple example, suppose we have the same exchangeable data from earlier (see Figure 6.1). Suppose we want to model the underlying distribution as a mixture of normal distributions, where each observation will have a mean μ_i and variance σ_i^2. By drawing these parameters from a distribution G where we place a DP prior on G we induce clustering of them, due to realisations from the DP being discrete distributions and hence there being a positive probability that some of these parameters will be identical. The clustering

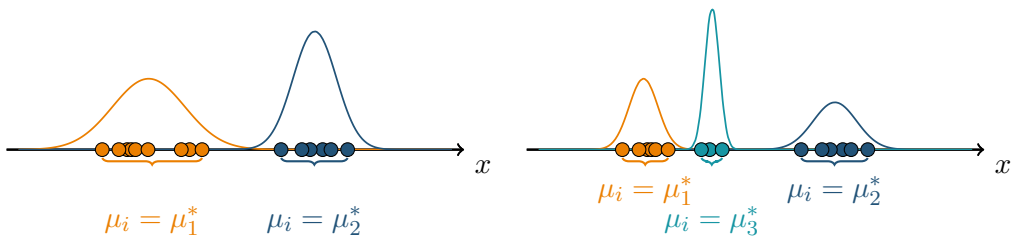

Figure 6.5: Possible clustering of observations using a DP mixture

of the means μ_i and variances σ_i^2 therefore induces a clustering of the observations. This model for these data can be written as

$$\begin{aligned} X_i|\mu_i, \sigma_i^2 &\sim N(\mu_i, \sigma_i^2) \qquad i = 1, \ldots, n \\ (\mu_i, \sigma_i^2)|G &\sim G \\ G &\sim DP(\alpha, G_0). \end{aligned}$$

The concentration parameter α determines the level of clustering, where larger values result in more clusters and the realisations from the process tend towards the base distribution G_0. The base distribution G_0 must have the same support as the process, and so in our scenario we need a base distribution for the means and variances. We take a normal-inverse-gamma distribution. Figure 6.5 demonstrates how different clusterings could be appropriate for the same data as seen in Figure 6.1 and how the DP mixture can vary in the number of clusters. In the first graphic the DP mixture assigns each data point to one of two clusters, whereas in the second graphic, the DP mixture finds three clusters to be more appropriate and the variance and means of these clusters will change accordingly, thus demonstrating the flexibility of the DP mixture - we have not specified the number of clusters to use.

6.4.1 Fitting through MCMC

Inference for Dirichlet process mixture models can be achieved through MCMC; see for example MacEachern and Müller (1998). Neal (2000) gives an overview of a number of algorithms that can be used, some of which are more desirable than others in terms of efficiency.

The final algorithm in Neal (2000) is one of the more popular approaches as it has been shown to be one of the most efficient sampling methods for Dirichlet Process mixtures (Papaspiliopoulos and Roberts, 2008). We will discuss this algorithm due to its efficiency and ability to work with non-conjugate priors. The algorithm uses a Gibbs sampling approach and enhances mixing through the use of auxiliary variables, of which there

are m. These auxiliary variables are the parameters of additional clusters to represent components without any observations assigned. The Markov chain will consist of cluster indicator variables c_i for each data point $\boldsymbol{x}_i$ and also the cluster-specific parameters $\boldsymbol{\phi}_c$.

For each iteration of the scheme, the algorithm loops over the data $\boldsymbol{x}_i$ for $i = 1, \ldots, n$ and allocates each one in turn given the current allocation of the rest. Using similar notation to allocate observation $\boldsymbol{x}_i$, we let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. If the current observation is a singleton (in a cluster by itself, that is $c_i \neq c_j$ for all $j \neq i$) then we let the first auxiliary variable have the value $\boldsymbol{\phi}_{c_i}$ and draw the rest from the base distribution G_0. If the observation is currently in a cluster with other observations (that is $c_i = c_j$ for some $j \neq i$) then we draw all m auxiliary variables from the base distribution G_0. The existing clusters are labelled $\{1, \ldots, k^-\}$ and the auxiliary clusters are labelled $\{k^- + 1, \ldots, h\}$. The observation is then allocated to a cluster by sampling its cluster indicator from a discrete distribution with probabilities

$$Pr(c_i = c|\boldsymbol{c}_{-i}, \boldsymbol{x}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{k^-+l}) \propto \begin{cases} \dfrac{n_{-i,c}}{n-1+\alpha} f(\boldsymbol{x}_i|\boldsymbol{\phi}_c, \boldsymbol{x}_c) & \text{for } 1 \leq c \leq k^- \\ \dfrac{\alpha/m}{n-1+\alpha} f(\boldsymbol{x}_i|\boldsymbol{\phi}_c) & \text{for } k^- + 1 \leq c \leq h \end{cases},$$

where $n_{-i,c}$ is the number of observations currently assigned to cluster c excluding observation i. We then delete any cluster that has no data assigned. Once all of the observations have been allocated to a cluster, we simply update the cluster-specific parameters conditional on the data assigned to them using appropriate Gibbs sampling or Metropolis-Hastings steps. The algorithm can be summarised in Algorithm 6.

6.4.2 Inferring the concentration parameter α

The concentration parameter α can be inferred from the data along with the parameters of the DP as shown in Escobar and West (1995). It can be shown that the posterior for α only depends on the distinct number of clusters in use k, that is

$$\pi(\alpha|\boldsymbol{c}, \boldsymbol{x}, \boldsymbol{\phi}) = \pi(\alpha|k).$$

If we suppose that the prior density for α is a Gamma prior, $\alpha \sim Ga(a, b)$, Escobar and West (1995) show that through use of an auxiliary Beta random variable η, the posterior for α can be expressed as a mixture of two Gamma densities. Thus, to sample from the posterior for α, we would sample $\eta \sim Beta(\alpha + 1, n)$. We then calculate the weights of

Algorithm 6 Algorithm 8 of Neal (2000)

Initialise. Let the Markov chain consist of cluster indicator variables c_i for each data point $\boldsymbol{x}_i, i = 1 \ldots, n$ and also the cluster-specific parameters $\boldsymbol{\phi}_c$. Repeatedly sample as follows:

- For $i = 1 \ldots, n$
  1. Let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. Label these values in $\{1, \ldots, k^-\}$.
  2. If $c_i = c_j$ for some $j \neq i$, draw values independently from the base distribution G_0 for those $\boldsymbol{\phi}_c$ for which $k^- < c \leq h$.
  3. If $c_i \neq c_j$ for all $j \neq i$, let c_i have the label $k^- + 1$, and draw values independently from the base distribution G_0 for those $\boldsymbol{\phi}_c$ for which $k^- + 1 < c \leq h$.
  4. Draw a new value for c_i from $\{1, \ldots, h\}$ using probabilities

$$Pr(c_i = c | \boldsymbol{c}_{-i}, \boldsymbol{x}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{k^-+l}) \propto \begin{cases} \dfrac{n_{-i,c}}{n-1+\alpha} f(\boldsymbol{x}_i | \boldsymbol{\phi}_c, \boldsymbol{x}_c) & \text{for } 1 \leq c \leq k^- \\ \dfrac{\alpha/m}{n-1+\alpha} f(\boldsymbol{x}_i | \boldsymbol{\phi}_c) & \text{for } k^- + 1 \leq c \leq h \end{cases},$$

     where $n_{-i,c}$ is the number of observations currently assigned to cluster c excluding observation i. We then delete any cluster that has no data assigned.
- For $c = c_1 \ldots, c_n$, update the cluster-specific parameters using Bayes' theorem.
- Return to step 1.

the two Gamma densities by

$$\frac{\pi_\eta}{(1-\pi_\eta)} = \frac{a+k-1}{n(b-\log\eta)}.$$

We then sample α from its full conditional

$$\alpha | \eta, k \sim \pi_\eta Ga(a+k, b-\log\eta) + (1-\pi_\eta) Ga(a+k-1, b-\log\eta).$$

A specific choice of prior for α is an implicit prior on the number of components k. From Antoniak (1974), the prior distribution on k may be written as

$$\pi(k|\alpha, n) = C_n(k) n! \alpha^k \frac{\Gamma(\alpha)}{\Gamma(\alpha+n)}, \qquad k = 1, 2, \ldots, n,$$

where $C_n(k) = \pi(k|\alpha = 1, n)$ thus does not involve α and can be calculated using recurrence formulae for Stirling numbers. Using simulation and integrating over αs we can then obtain prior probabilities for the number of clusters k for a specific prior on α.

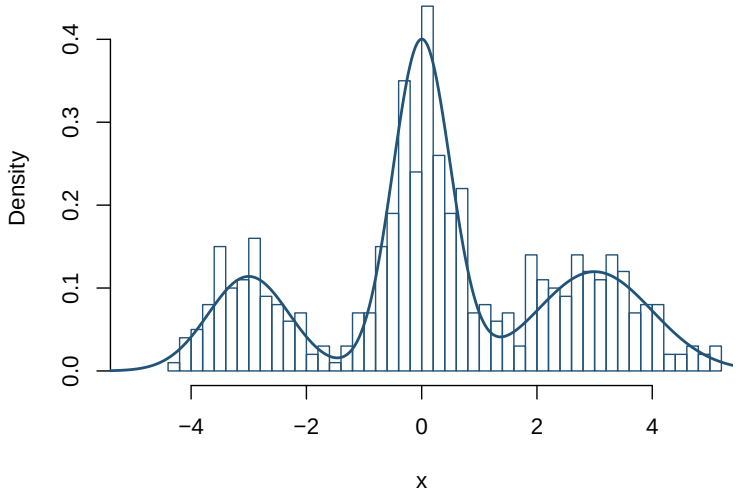

Figure 6.6: Histogram of data obtained from a three-component normal mixture with 'true' density overlaid (blue).

6.4.3 Example: Three-component normal mixture

We now look at an example of fitting a Dirichlet process mixture model to data obtained from a three-component normal mixture. This will demonstrate the flexibility of the model and how we do not need to pre-specify the number of components to use, something that can be quite difficult to do when modelling mixtures.

Following on from our earlier illustration where we modelled the unknown distribution as a mixture of normals with unknown mean and variance, we will perform a simulation study by generating some data from a three-component mixture of normals and then fit a Dirichlet process mixture model using the MCMC algorithm outlined above. The distribution we simulate our data from has the corresponding density function

$$f(x) = 0.2N(x; -3, 0.7^2) + 0.5N(x; 0, 0.5^2) + 0.3N(x; 3, 1^2),$$

and we draw $n = 500$ realisations from the associated distribution. The data are given in Figure 6.6 where we see three distinct peaks and the 'true' density is overlaid.

The means and variances of each data point will be drawn from a Dirichlet process and hence we need a suitable base distribution. A normal-inverse-gamma (NIG) prior is an appropriate choice of prior here as it will allow for a conjugate update of the parameters. That is

$$\sigma^2 \sim IG(a_\sigma, b_\sigma) \qquad \text{and} \qquad \mu|\sigma^2 \sim N(\mu_0, \sigma^2\lambda_0),$$

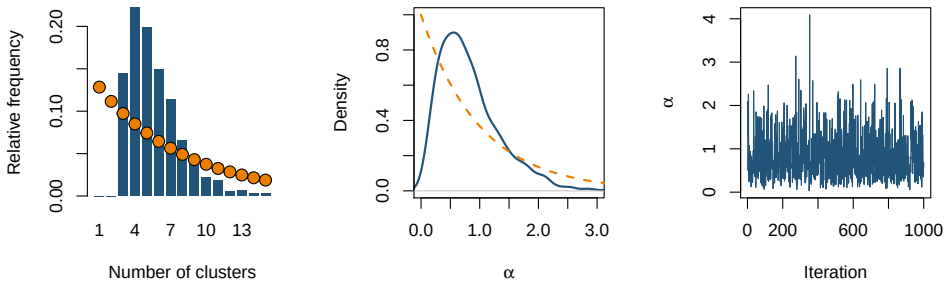

Figure 6.7: Posterior probabilities for the number of clusters (left) with prior probabilities overlaid in orange, marginal posterior density for α (middle) with prior density overlaid in orange and trace plot for α (right).

where $a_\sigma = 1, b_\sigma = 1, \mu_0 = 0$ and $\lambda_0 = 1$ are the hyperparameters. The Dirichlet process mixture model is defined as

$$\begin{aligned} x_i|\mu_i, \sigma_i^2 &\sim N(\mu_i, \sigma_i^2) \qquad i = 1, \ldots, n \\ (\mu_i, \sigma_i^2)|G &\sim G \\ G &\sim DP(\alpha, G_0), \\ G_0 &= NIG(\mu_0, \lambda_0, a_\sigma, b_\sigma), \\ \alpha &\sim Ga(a_\alpha, b_\alpha). \end{aligned}$$

Using MCMC we can fit the model using Algorithm 7. After 100 burn-in iterations are removed, every 100^{th} iterate is stored (to reduce autocorrelation between the samples) to give us 1K samples from the posterior. Figure 6.7 shows the posterior probabilities for the number of clusters as well as the posterior density for the concentration parameter. The distribution of the number of clusters is quite varied here, suggesting the data can be plausibly described by more than the 3 components used to simulate the data.

Samples from the posterior predictive distribution can be obtained using methods described in Section 6.3.5, where we draw $x^* \sim N(\mu^*, \sigma^{*2})$. The pair (μ^*, σ^{*2}) will be drawn from the base distribution with probability proportional to α or from the empirical distribution $\sum_{i=1}^n \delta_{(\mu_i, \sigma_i^2)}$ for each MCMC iterate. Figure 6.8 shows a histogram of samples from the predictive distribution.

6.4.4 Example: Two-component regression mixture

In this simulation study we simulate data from a two-component linear regression mixture with equal proportions. We will then fit a Dirichlet process mixture to the data which will then indicate how many components are required. For example, we may find that

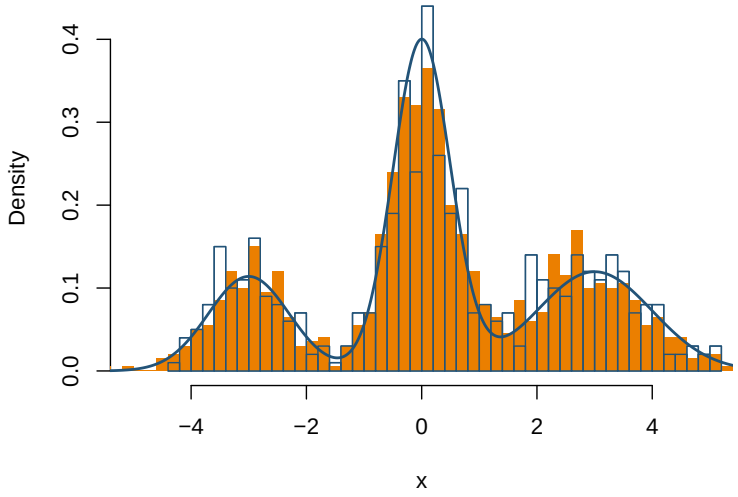

Figure 6.8: Samples from the posterior predictive distribution (orange) overlaid on to the histogram of the data with the 'true' density overlaid in blue.

a single component, corresponding to a simple linear regression model, could model the data adequately.

We took the input space to be a grid of $n = 500$ points in one-dimensional θ space between -1 and 1 giving $p = 2$ regression coefficients, $\boldsymbol{\beta} = (\beta_0, \beta_1)^\top$. We wish to model the output $X(\theta)$ as a linear function $\beta_0 + \beta_1\theta$. With equal probability each simulated point will be obtained from one of the two components where component i is

$$X_i(\theta_j) = \beta_0^i + \beta_1^i\theta_j + \epsilon_j^i, i = 1, 2$$

with $\epsilon_j^i \overset{indep}{\sim} N(0, \sigma_i^2)$. The first component has regression line with $\beta_0^1 = 1, \beta_1^1 = 0.5$ and noise $\sigma_1^2 = 0.1$ and the other has the parameters $\beta_0^2 = -1, \beta_1^2 = 1$ with noise $\sigma_2^2 = 0.05$. The data are shown in Figure 6.9.

The regression coefficients and error variance will be drawn from the Dirichlet process and hence we need a suitable base distribution. A normal-inverse-gamma is an appropriate choice again. That is

$$\sigma^2 \sim IG(a_\sigma, b_\sigma) \qquad \text{and} \qquad \boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{\mu}_\beta, \sigma^2 V_\beta),$$

where $a_\sigma, b_\sigma, \boldsymbol{\mu}_\beta$ and V_β are the hyperparameters.

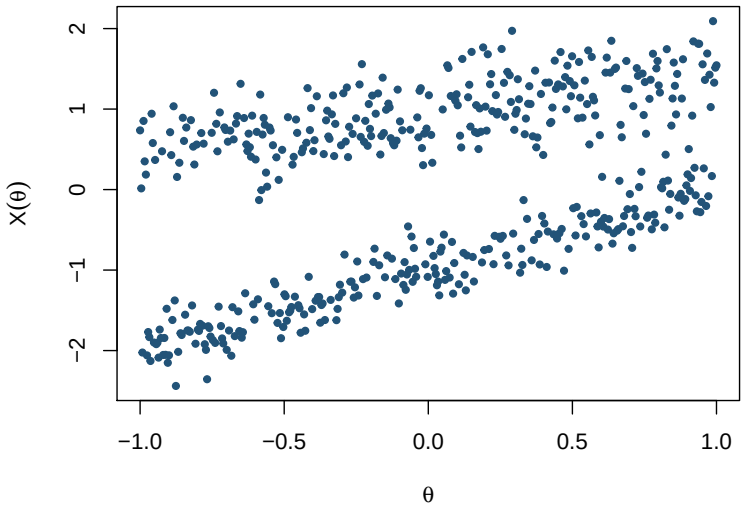

Figure 6.9: Data obtained from a two-component regression mixture.

We then have the following model, which we fit using Algorithm 8,

$$
\begin{aligned}
X(\theta_i)|\boldsymbol{\beta}_i, \sigma_i^2 &\sim N(\beta_{i0} + \beta_{i1}\theta_i, \sigma_i^2) \qquad i = 1, \ldots, n \\
(\boldsymbol{\beta}_i, \sigma_i^2)|G &\sim G \\
G &\sim DP(\alpha, G_0) \\
G_0 &= NIG(\boldsymbol{\mu}_\beta, V_\beta, a_\sigma, b_\sigma) \\
\alpha &\sim Ga(a_\alpha, b_\alpha).
\end{aligned}
$$

After a small burn-in period of 100 iterations, the chain looks to have converged as can be seen in any of the trace plots in Figures 6.11 to 6.14. Keeping every 10^{th} iterate (to reduce autocorrelation), we obtain 1K samples from the posterior. Figures 6.10 show the posterior probabilities for the number of clusters as well as the posterior density for the concentration parameter. We can see the modal number of clusters is 2, which is what we expected, with the data informing us that there must be at least two clusters since we have zero probability of one cluster in this case.

Using the samples from the posterior distribution, we can draw values from the predictive distribution as a way to visualise the fitted mixture model. Figure 6.15 shows samples from the predictive fit at a number of new points in θ space. It is evident that the resulting mixture has picked out the two components in the simulated data without the need to pre-specify the number of components we wanted to fit, thus demonstrating the flexibility of a Dirichlet process mixture model.

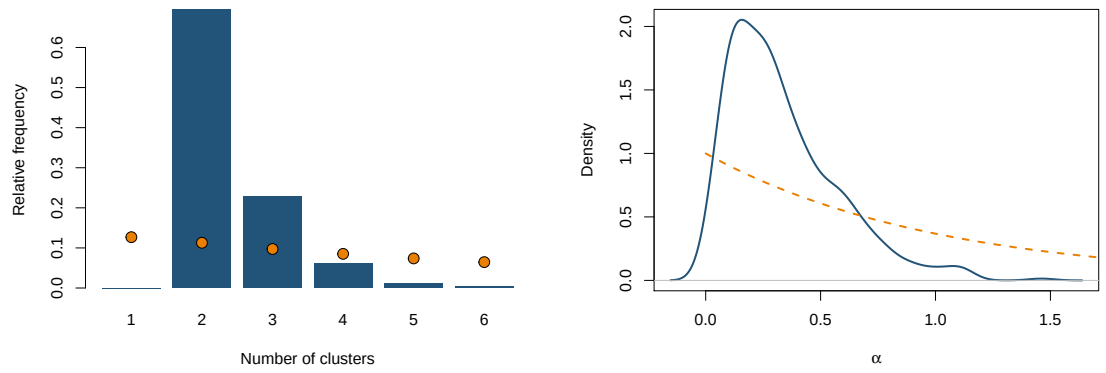

Figure 6.10: Posterior probabilities for the number of clusters (left) with prior probabilities overlaid in orange and marginal posterior density for α (right) with prior density overlaid in orange.

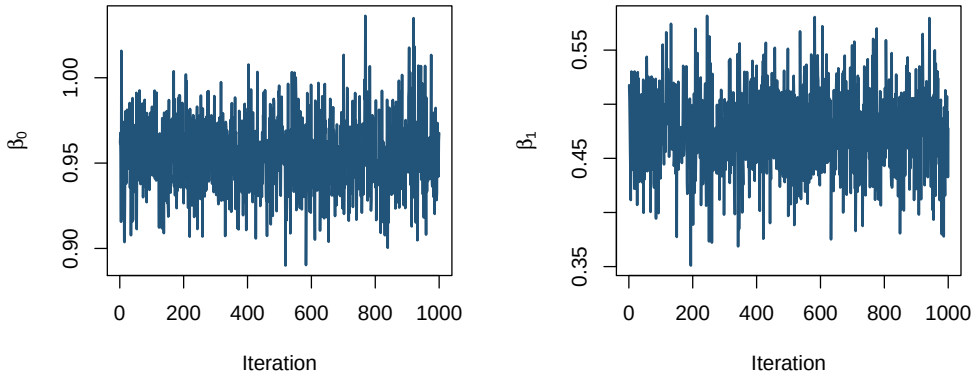

Figure 6.11: Trace plot of posterior samples for β_0 and β_1 for the first cluster.

We could easily imagine this data as being some multi-modal stochastic model output at each point in θ space at one point in time. Of course, the stochastic nature of the biological models discussed in Chapter 2 would mean we should really have replicate observations at each point and typically we would need to use something more complex than simple linear regression. We now expand on this idea in the next chapter.

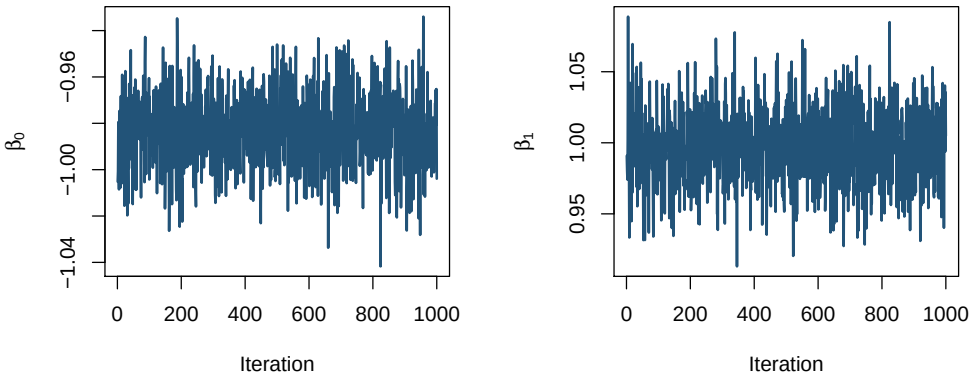

Figure 6.12: Trace plot of posterior samples for β_0 and β_1 for the second cluster.

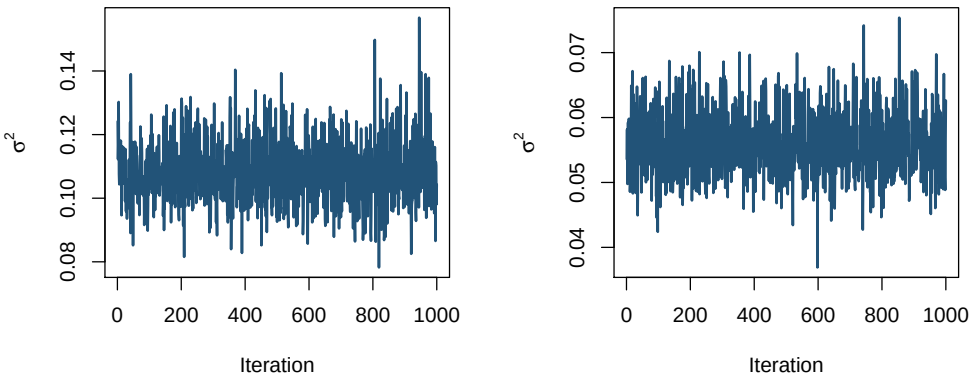

Figure 6.13: Trace plot of posterior samples for σ^2 for the first (left) and second (right) cluster.

Algorithm 7 MCMC algorithm to fit a DPM of univariate normals
(using Algorithm 8 of Neal (2000))

Initialise. Let the Markov chain consist of cluster indicator variables c_i for each data point $x_i, i = 1 \ldots, n$ and also the cluster-specific parameters $\boldsymbol{\phi}_c = (\mu_c, \sigma_c^2)^\top$. Repeatedly sample as follows:

- For $i = 1 \ldots, n$
  1. Let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. Label these values in $\{1, \ldots, k^-\}$.
  2. If $c_i = c_j$ for some $j \neq i$, draw values independently from the base distribution $G_0 = NIG(\mu_0, \lambda_0, a_\sigma, b_\sigma)$ for those $\boldsymbol{\phi}_c$ for which $k^- < c \leq h$.
  3. If $c_i \neq c_j$ for all $j \neq i$, let c_i have the label k^-+1, and draw values independently from the base distribution $G_0 = NIG(\mu_0, \lambda_0, a_\sigma, b_\sigma)$ for those $\boldsymbol{\phi}_c$ for which $k^- + 1 < c \leq h$.
  4. Draw a new value for c_i from $\{1, \ldots, h\}$ using probabilities

$$Pr(c_i = c | \boldsymbol{c}_{-i}, \boldsymbol{x}, \boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{k^-+l}) \propto \begin{cases} \dfrac{n_{-i,c}}{n-1+\alpha} N(\boldsymbol{x}_i; \mu_c, \sigma_c^2) & \text{for } 1 \leq c \leq k^- \\ \dfrac{\alpha/m}{n-1+\alpha} N(\boldsymbol{x}_i; \mu_c, \sigma_c^2) & \text{for } k^- + 1 \leq c \leq h \end{cases},$$

     where $n_{-i,c}$ is the number of observations currently assigned to cluster c excluding observation i. We then delete any cluster that has no data assigned.

- For $c = c_1 \ldots, c_n$, update the cluster-specific parameters using Bayes' Theorem and drawing from the posteriors.
  1. Draw $\sigma_c^2 \sim IG(a_\sigma^*, b_\sigma^*)$
  2. Draw $\mu_c | \sigma_c^2 \sim N(\mu_0^*, \sigma_c^2 \lambda_0^*)$
- Sample α as in Section 6.4.2.
- Return to step 1.

Algorithm 8 MCMC algorithm to fit a DPM of linear regression models (using Algorithm 8 of Neal (2000))

Initialise. Let the Markov chain consist of cluster indicator variables c_i for each data point $x_i, i = 1 \ldots, n$ and also the cluster-specific parameters $\boldsymbol{\phi}_c = (\boldsymbol{\beta}_c, \sigma_c^2)^\top$. Repeatedly sample as follows:

- For $i = 1 \ldots, n$
  1. Let k^- be the number of distinct c_j for $j \neq i$ and let $h = k^- + m$. Label these values in $\{1, \ldots, k^-\}$.
  2. If $c_i = c_j$ for some $j \neq i$, draw values independently from the base distribution $G_0 = NIG(\boldsymbol{\mu}_\beta, V_\beta, a_\sigma, b_\sigma)$ for those $\boldsymbol{\phi}_c$ for which $k^- < c \leq h$.
  3. If $c_i \neq c_j$ for all $j \neq i$, let c_i have the label $k^- + 1$, and draw values independently from the base distribution $G_0 = NIG(\boldsymbol{\mu}_\beta, V_\beta, a_\sigma, b_\sigma)$ for those $\boldsymbol{\phi}_c$ for which $k^- + 1 < c \leq h$.
  4. Draw a new value for c_i from $\{1, \ldots, h\}$ using probabilities

$$Pr(c_i = c | \boldsymbol{c}_{-i}, \cdot) \propto \begin{cases} \dfrac{n_{-i,c}}{n - 1 + \alpha} N(\boldsymbol{x}_i; \beta_{0_c} + \beta_{1_c}\theta_i, \sigma_c^2) & \text{for } 1 \leq c \leq k^- \\ \dfrac{\alpha/m}{n - 1 + \alpha} N(\boldsymbol{x}_i; \beta_{0_c} + \beta_{1_c}\theta_i, \sigma_c^2) & \text{for } k^- + 1 \leq c \leq h \end{cases},$$

     where $n_{-i,c}$ is the number of observations currently assigned to cluster c excluding observation i. We then delete any cluster that has no data assigned.

- For $c = c_1 \ldots, c_n$, update the cluster-specific parameters using Bayes' Theorem and drawing from the posteriors.
  1. Draw $\sigma_c^2 \sim IG(a_\sigma^*, b_\sigma^*)$
  2. Draw $\mu_c | \sigma_c^2 \sim N(\boldsymbol{\mu}_\beta^*, \sigma_c^2 V_\beta^*)$
- Sample α as in Section 6.4.2.
- Return to step 1.

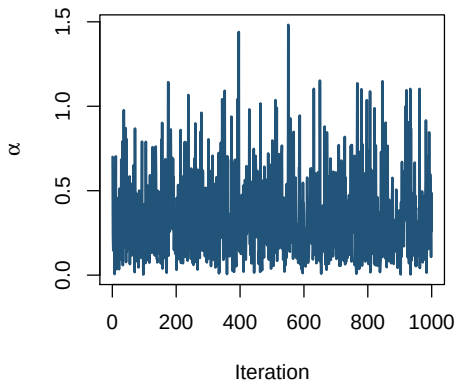

Figure 6.14: Trace plot of posterior samples for α.

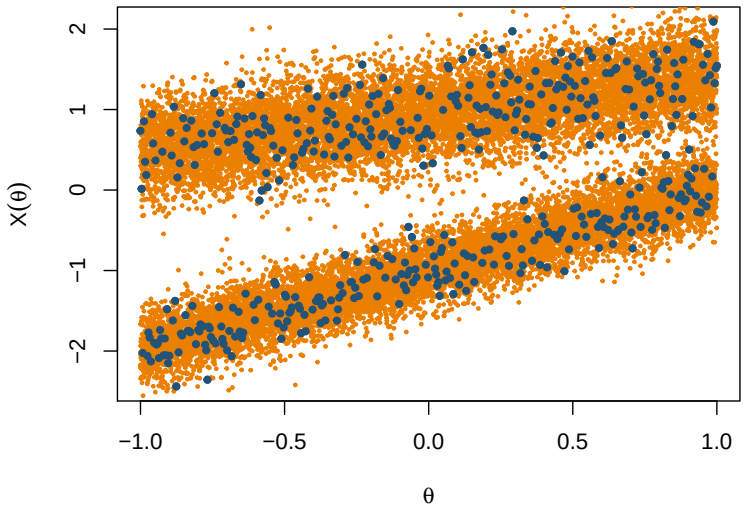

Figure 6.15: Samples from the posterior predictive distribution (orange) with data overlaid (blue).

Chapter 7

Dirichlet process mixture of Gaussian processes

7.1 Introduction

In Chapter 2 we introduced stochastic kinetic models and in Chapter 3 explained how we might perform inference for the parameters of these models, the stochastic rate constants, through the use of likelihood-free methods.

These likelihood-free methods require us to be able to simulate from the model, something which can become very demanding for complex models with a large number of reactions and species. This suggests the use of an approximate model, or emulator, to be used in place of the expensive simulator.

In Chapter 5 we explained how Gaussian processes can be used as an emulator by considering the spatial correlation between training inputs. Placing particular emphasis on the Schlögl model, we require a mixture of Gaussian processes. In Chapter 6 we explained the concept of mixture modelling, defined Dirichlet processes and demonstrated how they can be used in mixture modelling through two examples.

In this chapter we construct the model which we will use to emulate SKM output by combining the concepts of Dirichlet processes and Gaussian processes to create a Dirichlet process mixture of Gaussian processes. Moreover, we provide details of how to fit the resulting emulator to training data generated from the SKM simulator.

7.2 Adding spatial correlation

We require a flexible model of the output from a generic stochastic kinetic model. The previous Dirichlet process mixture (DPM) of normals isn't appropriate since there is no dependence between the log rates $\boldsymbol{\theta}$. We need to build this spatial dependence in to the emulator. Figure 7.1 depicts this spatial dependence, under the simplifying assumption of two-dimensional $\boldsymbol{\theta}$ space, and shows three $\boldsymbol{\theta}$ values at which predicted output may be required. For proposed rates $\tilde{\boldsymbol{\theta}}$, we need to approximate the distribution $X(\tilde{\boldsymbol{\theta}})$ at this new location in $\boldsymbol{\theta}$ space. The output here should be highly correlated with the output of nearby locations. Hence if we train our emulator with enough simulator output (black points), we can use prediction to find an accurate approximate distribution at new points (orange points).

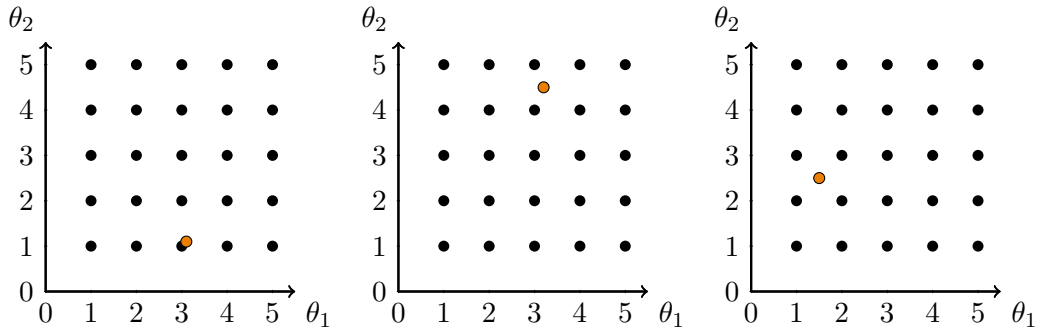

Figure 7.1: Grid of points in $\boldsymbol{\theta}$ space where observations are obtained and three example proposed $\tilde{\boldsymbol{\theta}}$ points (orange) at which $X(\tilde{\boldsymbol{\theta}})$ is required.

The simulator output that we train our emulator with will be built up with M replicates at each of the N locations in d dimensional $\boldsymbol{\theta}$ space. These points are chosen in order to optimise the spread of the locations in d dimensional $\boldsymbol{\theta}$ space which is achieved by using a maximin Latin hypercube design (Section 5.2). If we were trying to emulate a deterministic process then there would be no need for replicate observations at each point in space since the output would be identical for the same input. However, since the process is both stochastic and possibly multi-modal (e.g. Schlögl system) then the replicate observations are necessary if we want the emulator to capture the marginal distribution correctly at each point in the input space.

We begin with the simpler problem of how to build the emulator at only a single point in time and suppress any dependence on t. The emulator will then be fit, independently, at each time point where we have data. The training data, that is the data we use to train the emulator at each time point, is then the collection of observations

$$\boldsymbol{X} = (X_1(\boldsymbol{\theta}_1), \ldots, X_M(\boldsymbol{\theta}_1), \ldots, X_1(\boldsymbol{\theta}_N), \ldots, X_M(\boldsymbol{\theta}_N))^\top.$$

We could fit a Gaussian process (GP) to this data if we assumed the output was marginally normal, however, recalling the Schlögl system in Chapter 2, we wish to have a more flexible emulator that will be able to adequately model mulit-modal output. We therefore consider a mixture of GPs which would allow us to model the bimodal behaviour of the Schlögl system. A finite mixture of say two components might be sufficient but we can make use of Dirichlet process mixture modelling as discussed in Chapter 6 which means we don't need to specify the number of components *a priori*. Thus, we build a Dirichlet process mixture of Gaussian processes (DPMGP) as follows

$$
\begin{aligned}
X_i(\boldsymbol{\theta}_j)|\boldsymbol{\beta},\sigma^2,\boldsymbol{r},\nu &\overset{indep}{\sim} N\{m(\boldsymbol{\theta}_j),K(\boldsymbol{\theta}_j)\} \qquad i=1,\ldots,M, \quad j=1,\ldots,N\\
\boldsymbol{\beta},\sigma^2,\boldsymbol{r},\nu|G &\sim G\\
G|\alpha,G_0 &\sim DP(\alpha,G_0)\\
G_0 &= N_pIG(\boldsymbol{m}_0,V_0,a_\sigma,b_\sigma)\times LN_d(\boldsymbol{a_r},b_{\boldsymbol{r}})\times IG(a_\nu,b_\nu)\\
\alpha &\sim Ga(a_\alpha,b_\alpha)
\end{aligned}
$$

where $\boldsymbol{\beta},\sigma^2,\boldsymbol{r},\nu$ are all GP hyperparameters. This model allows each observation $X_i(\boldsymbol{\theta}_j)$ to be clustered individually for $i=1,\ldots,M, j=1,\ldots,N$ which is necessary when fitting to SKM output since each draw from the simulator will be independent.

If we look at the model hierarchy we can see that it is the parameters of the GP mean and covariance functions that are drawn from the DP, not the actual GP functions. This induces clustering of the observations, essentially assigning each to a GP. We use the typical choice of correlation function for each GP, the squared exponential function where, for GP k, we have

$$
(H_k)_{ij}=\exp\left\{-\sum_{l=1}^{d}(\theta_{il}-\theta_{jl})^2/r_{kl}^2\right\}. \tag{7.1}
$$

The covariance matrix for GP k is then $K_k=\sigma_k^2(\nu_kI_{n_k}+H_k)$ where n_k is the total number of observations assigned to cluster k and we add a nugget parameter ν_k for numeric stability and to act as a noise model for the observations.

If we assume that the prior mean function is linear in the d parameters $\boldsymbol{\theta}_i$ and there is one replicate at each point in space then the design matrix $\mathcal{X}$ and parameter vector $\boldsymbol{\beta}$ is

$$
\mathcal{X}=\begin{pmatrix}1 & \theta_{11} & \cdots & \theta_{d1}\\ 1 & \theta_{12} & \cdots & \theta_{d2}\\ \vdots & \vdots & \ddots & \vdots\\ 1 & \theta_{1N} & \cdots & \theta_{dN}\end{pmatrix} \qquad \text{and} \qquad \boldsymbol{\beta}_k=\begin{pmatrix}\beta_{k0}\\ \beta_{k1}\\ \vdots\\ \beta_{kd}\end{pmatrix}. \tag{7.2}
$$

The base distribution is then a combination of independent distributions over these hy-

perparameters, where we place a normal-inverse-gamma prior on the coefficients of the mean function and scale, a lognormal prior on the components of the correlation lengths $\boldsymbol{r}_k$ and an inverse gamma prior on the nugget parameters ν_k.

There are examples of similar models in the literature, particulary in Biology (McDowell et al. (2018), Hensman et al. (2014), Cooke et al. (2011)). These models are similar to our approach as the GP hyerparameters are drawn from the DP, however we will have replicate observations at each point in $\boldsymbol{\theta}$ space and thus we can exploit this to produce some novel ways of reducing the computational cost of fitting as demonstrated below.

7.3 Evaluation of the log-likelihood function

The log-likelihood function is the sum of the cluster-specific log-likelihood functions. Each cluster log-likelihood function involves calculating the determinant and inverse of a potentially large covariance matrix $K_k = \sigma_k^2(\nu_k I_{n_k} + H_k)$, where H_k is the $n_k \times n_k$ correlation matrix calculated using the squared exponential covariance function, that is, has elements as in (7.1) and n_k is the total number of observations assigned to cluster k. The size of K_k can lead to numerical instabilities in the calculation of its inverse and determinant as well as incurring considerable computational expense. However, these quantities can be determined via calculations on much smaller matrices due to their block structure.

We assume that the number of (input) $\boldsymbol{\theta}$–points with observations assigned to the cluster is N and each $\boldsymbol{\theta}_i$ has n_i replicates assigned to the cluster. The total number of observations assigned to the cluster is $n_k = \sum_i n_i$. The derivation will also make use of an $n_k \times N$ "design" matrix J which indicates which observations (rows) correspond to which $\boldsymbol{\theta}$ value

(columns), where

$$J = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n_1 \text{ rows} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n_2 \text{ rows} \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\} n_N \text{ rows} \end{matrix} \tag{7.3}$$

The mean vector for the cluster will then be of length n_k and can be written as $J\mathcal{X}\boldsymbol{\beta}$. In the following derivations we drop the dependence on cluster number for ease of exposition, but in practice, the following quantities would have a subscript k to denote that they are calculated for cluster k.

7.3.1 Inverse of the correlation matrix

The covariance matrix is $K = \sigma^2(\nu I_n + H)$ and the inverse of its correlation matrix is determined as follows. First we can write

$$(\nu I_n + H)^{-1} = (\nu I_n + J H_N J^\top)^{-1},$$

where J is as in (7.3). Here the $N \times N$ matrix H_N is a correlation matrix between the realisations at distinct $\boldsymbol{\theta}_i$ in the cluster. Using the Woodbury identity (Hager, 1989)

$$(A + CBD)^{-1} = A^{-1} - A^{-1}C(B^{-1} + DA^{-1}C)^{-1}DA^{-1} \tag{7.4}$$

we have

$$\begin{aligned} (\nu I_n + J H_N J^\top)^{-1} &= \nu^{-1} I_n - \nu^{-1} J (H_N^{-1} + \nu^{-1} J^\top J)^{-1} \nu^{-1} J^\top \\ &= \nu^{-1} I_n - \nu^{-2} J (H_N^{-1} + \nu^{-1} J^\top J)^{-1} J^\top. \end{aligned}$$

Now $D = J^\top J = \text{diag}(n_1, n_2, \ldots, n_N)$ is an $N \times N$ diagonal matrix whose diagonal elements are the number of observations for each $\boldsymbol{\theta}_i$ in the cluster. Hence

$$\begin{aligned}(\nu I_n + H)^{-1} &= \nu^{-1} I_n - \nu^{-2} J(H_N^{-1} + \nu^{-1} D)^{-1} J^\top \\ &= \nu^{-1} I_n - \nu^{-2} J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top.\end{aligned}$$

This result allows us to find the inverse of a large $n \times n$ matrix by only calculating the inverse of a smaller $N \times N$ matrix, thus being more computationally efficient.

7.3.2 Determinant of the correlation matrix

Since the correlation matrix can be written as $\nu I_n + J H_N J^\top$, its determinant can be simplified using Sylvester's determinant identity (Sylvester, 1851): for matrices A $(m \times n)$ and B $(n \times m)$, we have $\det(I_m + AB) = \det(I_n + BA)$. Hence, for our problem we have that

$$\begin{aligned}\det(\nu I_n + J H_N J^\top) &= \nu^n \det(I_n + \nu^{-1} J H_N J^\top) \\ &= \nu^n \det(I_N + \nu^{-1} J^\top J H_N) \\ &= \nu^n \det(I_N + \nu^{-1} D H_N).\end{aligned}$$

Therefore the determinant of the $n \times n$ correlation matrix can be calculated in terms of the determinant of an $N \times N$ matrix, again offering computational savings.

7.3.3 Log-likelihood function

Using the above methods we can now efficiently evaluate the log-likelihood function (with the full derivation in Appendix A.3).

The data is stacked as $\underline{\boldsymbol{x}} = (x_1(\theta_1), x_2(\theta_1), \ldots, x_{n_1}(\theta_1), \ldots, x_1(\theta_N), \ldots, x_{n_N}(\theta_N))^\top = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top$ where $\boldsymbol{x}_i$ is a $n_i \times 1$ vector of all observations assigned to the cluster at $\boldsymbol{\theta}_i$. The log-likelihood contribution of the cluster is given by (up to an additive constant)

$$\begin{aligned}&-2 \log f\{\boldsymbol{x}_1(\theta_1), \ldots, \boldsymbol{x}_N(\theta_N)\} \\ &\quad = 2n \log \sigma + n \log \nu + \log |I_N + \nu^{-1} D H_N| \\ &\qquad + \frac{1}{\nu \sigma^2} \Bigg\{ (\underline{\boldsymbol{x}} - JX\boldsymbol{\beta})^\top (\underline{\boldsymbol{x}} - JX\boldsymbol{\beta}) \\ &\qquad\qquad - \nu^{-1} (\bar{\boldsymbol{x}} - X\boldsymbol{\beta})^\top D (I_N + \nu^{-1} H_N D)^{-1} H_N D (\bar{\boldsymbol{x}} - X\boldsymbol{\beta}) \Bigg\}.\end{aligned}$$

7.4 Assigning each point to a cluster

Implementing the Dirichlet process mixture is fairly straightforward using Algorithm 8 of Neal (2000) as previously discussed in Section 6.4.1. We will revisit the algorithm again now with the focus of using it for fitting the DPMGP model.

We sample each of the MN elements of the allocation vector $\boldsymbol{c}$ one at a time conditional on the rest as follows. Let $\boldsymbol{\Psi}_j = (\boldsymbol{\beta}_j, \sigma_j^2, \nu_j, \boldsymbol{r}_j)^\top$ be the cluster-specific parameters of the jth cluster and let x_i represent any one of the MN observations we wish to assign, for $i = 1, \ldots, MN$. Choosing a new value for c_i, that is allocating the ith observation to a cluster, we let k^- be the number of unique c_j for $j \neq i$. If the current value of $c_i = c_j$ for some $j \neq i$ then the cluster-specific parameters of the l auxiliary clusters are drawn from the base distribution G_0. If the ith observation is a singleton then the parameters of one of the l auxiliary clusters are assigned the values of the current cluster parameters of c_i and the remaining $l-1$ (for $l > 1$) auxiliary clusters have parameters drawn from the base distribution. The existing clusters with data assigned are labelled $\{1, \ldots, k^-\}$ and the auxiliary clusters are labelled $\{k^- + 1, \ldots, k^- + l\}$. We now draw a value for c_i from $\{1, \ldots, k^- + l\}$ using the probabilities

$$Pr(c_i = c | \boldsymbol{c}_{-i}, \boldsymbol{x}, \boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_{k^-+l}) \propto \begin{cases} \dfrac{n_{-i,c}}{MN - 1 + \alpha} f(x_i | \boldsymbol{\Psi}_c, \boldsymbol{x}_c) & \text{for } 1 \leq c \leq k^- \\ \dfrac{\alpha / l}{MN - 1 + \alpha} f(x_i | \boldsymbol{\Psi}_c) & \text{for } k^- + 1 \leq c \leq k^- + l \end{cases}$$

where $n_{-i,c}$ is the number of observations in cluster c excluding the ith observation and $\boldsymbol{x}_c$ is the data currently assigned to cluster c.

Suppose that observation x_i is the response at input $\boldsymbol{\theta}$. Then the density $f(x_i | \boldsymbol{\Psi}_c) = N\{x_i; \boldsymbol{\beta}_c^\top \boldsymbol{\theta}, (\nu_c + 1)\sigma_c^2\}$ and $f(x_i | \boldsymbol{\Psi}_c, \boldsymbol{x}_c)$ is the density of X_i further conditioned on the data $\boldsymbol{x}_c$ currently assigned to the cluster. The cluster has $n_{-i,c}$ observations assigned at a collection of $N_c \leq N$ points in $\boldsymbol{\theta}$–space which we shall denote $\boldsymbol{\Theta}_c$. Let $\boldsymbol{\theta}_{cj} = (\boldsymbol{\Theta}_c)_j$ for $j = 1, \ldots, N_c$ and suppose there are $n_{-i,cj}$ observations at each $\boldsymbol{\theta}_{cj}$–point in the cluster. The joint distribution of the data $\boldsymbol{x}_c$ in the cluster and observation x_i is a $(n_{-i,c} + 1)$-dimensional normal with mean

$$\big(\overbrace{m_c(\boldsymbol{\theta}_{c1}), \ldots, m_c(\boldsymbol{\theta}_{c1})}^{n_{-i,c1} \text{ elements}}, \ldots, \overbrace{m_c(\boldsymbol{\theta}_{cN_c}), \ldots, m_c(\boldsymbol{\theta}_{cN_c})}^{n_{-i,cN_c} \text{ elements}}, m_c(\boldsymbol{\theta})\big)^\top = \begin{pmatrix} J_{-i,c} X_c \boldsymbol{\beta}_c \\ m_c(\boldsymbol{\theta}) \end{pmatrix}$$

and covariance matrix

$$\sigma_c^2 \begin{pmatrix} \nu_c I_{n_{-i,c}} + J_{-i,c} H_{N_c} J_{-i,c}^\top & J_{-i,c} \boldsymbol{h}_c \\ \boldsymbol{h}_c^\top J_{-i,c}^\top & \nu_c + 1 \end{pmatrix},$$

where $\boldsymbol{h}_c$ is a $N_c \times 1$ vector containing the correlation between $x_i(\boldsymbol{\theta})$ and the observations in the cluster, with elements

$$(\boldsymbol{h}_c)_j = \exp\left\{-\sum_{l=1}^{d}(\theta_l - \theta_{cjl})^2/r_l^2\right\}.$$

Using standard multivariate normal results (A.2) the density $f(x_i|\boldsymbol{\Psi}_c, \boldsymbol{x}_c) = N(x_i; \mu_c^*, \sigma_c^{*2})$ has mean

$$\mu_c^* = \boldsymbol{\beta}_c^\top \boldsymbol{\theta} + \nu_c^{-1}\boldsymbol{h}_c^\top D_c\{I_{N_c} - \nu_c^{-1}(I_{N_c} + \nu_c^{-1}H_{N_c}D_c)^{-1}H_{N_c}D_c\}(\bar{\boldsymbol{x}}_c - X_c\boldsymbol{\beta}_c)$$

and variance

$$\sigma_c^{*2} = \sigma_c^2\left[\nu_c + 1 - \nu_c^{-1}\boldsymbol{h}_c^\top D_c\{I_{N_c} - \nu_c^{-1}(I_{N_c} + \nu_c^{-1}H_{N_c}D_c)^{-1}H_{N_c}D_c\}\boldsymbol{h}_c\right].$$

7.5 Prediction

Since we are using this model to build an emulator, prediction will play an important role in simulating from the distribution of the DPMGP at new input points. To predict $X(\tilde{\boldsymbol{\theta}})$ at a new point in $\boldsymbol{\theta}$ space we use the MCMC output of samples from the posterior distribution and take a random iteration to give us a (posterior) value for each of α, k and cluster-specific parameters $\boldsymbol{\Psi}_1^*, \ldots, \boldsymbol{\Psi}_k^*$, where $\boldsymbol{\Psi} = (\boldsymbol{\beta}, \sigma^2, \nu, \boldsymbol{r})^\top$. We then draw a cluster to assign $\tilde{\boldsymbol{\theta}}$ to from the following mixture

$$\pi(\tilde{\boldsymbol{\Psi}}|\cdot) \sim \frac{\alpha}{\alpha + MN}G_0(\tilde{\boldsymbol{\Psi}}) + \frac{1}{\alpha + MN}\sum_{i=1}^{k} n_i\delta_{\tilde{\boldsymbol{\Psi}}_i^*}$$

where n_i is the number of observations in cluster i. This results in either drawing a new cluster (with probability proportional to α) or an existing cluster (with probability proportional to n_i). Thus, larger clusters (in terms of n_i) will have a higher probability of being drawn than smaller clusters. The value for our realisation for $X(\tilde{\boldsymbol{\theta}})$ depends on the cluster realisation drawn above. If this is a new cluster then $X(\tilde{\boldsymbol{\theta}}) \sim N\{\tilde{\boldsymbol{\beta}}^\top\tilde{\boldsymbol{\theta}}, (\tilde{\nu}+1)\tilde{\sigma}^2\}$, where $(\tilde{\boldsymbol{\beta}}, \tilde{\sigma}^2, \tilde{\nu})$ is a realisation from the base distribution G_0. If this is an existing cluster, say cluster c, then the realisation has to be consistent with the GP in that cluster and so (similar to the fitted distribution in the previous section) $X(\tilde{\boldsymbol{\theta}}) \sim N(\tilde{\mu}, \tilde{\sigma}^2)$, where

$$\tilde{\mu} = \boldsymbol{\beta}_c^\top\tilde{\boldsymbol{\theta}} + \nu_c^{-1}\boldsymbol{h}_c^\top D_c\{I_{N_c} - \nu_c^{-1}(I_{N_c} + \nu_c^{-1}H_{N_c}D_c)^{-1}H_{N_c}D_c\}(\bar{\boldsymbol{x}}_c - X\boldsymbol{\beta}_c)$$

and

$$\tilde{\sigma}^2 = \sigma_c^2 \left[\nu_c + 1 - \nu_c^{-1} \boldsymbol{h}_c^\top D_c \{ I_{N_c} - \nu_c^{-1} (I_{N_c} + \nu_c^{-1} H_{N_c} D_c)^{-1} H_{N_c} D_c \} \boldsymbol{h}_c \right],$$

after suppressing the dependence of $X(\tilde{\boldsymbol{\theta}})$ on $\boldsymbol{\Psi}$.

7.6 Example: Two-component GP mixture

We will now look at an example of fitting this model to data simulated from a two-component mixture of Gaussian processes. The resulting predictive fit will demonstrate the flexibility of the model especially considering that the number of clusters does not need to be specified *a priori*.

The input space is chosen to be a grid of $N = 50$ points in one-dimensional θ space between -1 and 1 with $M = 50$ replicate observations at each point. We will use a linear mean function $m(\boldsymbol{\theta}) = \beta_0 + \beta_1 \boldsymbol{\theta}$ for both GPs. With equal probability each point will be obtained from one of two GPs independently. One GP has mean function coefficients $\beta_0 = -4, \beta_1 = 0.5$ with covariance kernel hyperparameters $\sigma^2 = 0.1, \nu = 6$ and correlation length $r = 0.1$ where we use the squared exponential covariance function

$$(K)_{ij} = \exp\{-(\theta_i - \theta_j)^2 / r^2\}.$$

The other GP has mean function coefficients $\beta_0 = 4, \beta_1 = 5$ with covariance kernel hyperparameters $\sigma^2 = 0.4, \nu = 4$ and correlation length $r = 0.1$. The data is shown in Figure 7.2.

The aim here is to fit the DPMGP model to these data and assess the predictive fit, since we could easily imagine SKM output varying through θ space in a similar manner. The model we fit is

$$\begin{aligned}
X_i(\boldsymbol{\theta}_j) | \boldsymbol{\beta}, \sigma^2, \boldsymbol{r}, \nu &\overset{indep}{\sim} N\{m(\boldsymbol{\theta}_j), K(\boldsymbol{\theta}_j)\} \qquad i = 1, \ldots, M, \quad j = 1, \ldots, N \\
\boldsymbol{\beta}, \sigma^2, \boldsymbol{r}, \nu | G &\sim G \\
G | \alpha, G_0 &\sim DP(\alpha, G_0) \\
G_0 &= N_p IG(\boldsymbol{m}_0, V_0, a_\sigma, b_\sigma) \times LN_d(\boldsymbol{a_r}, b_{\boldsymbol{r}}) \times IG(a_\nu, b_\nu) \\
\alpha &\sim Ga(a_\alpha, b_\alpha)
\end{aligned}$$

In this case, $p = 2, d = 1$ and we choose the following prior hyperparameters: $\boldsymbol{m}_0 = (0, 3)^\top, V_0 = 10 I_2, a_\sigma = 0.2, b_\sigma = 0.1, a_\nu = 1, b_\nu = 0.01, a_r = -1, b_r = 1, a_\alpha = 1$ and $b_\alpha = 1$, selected such that prior information is fairly weak. Upon fitting this model with

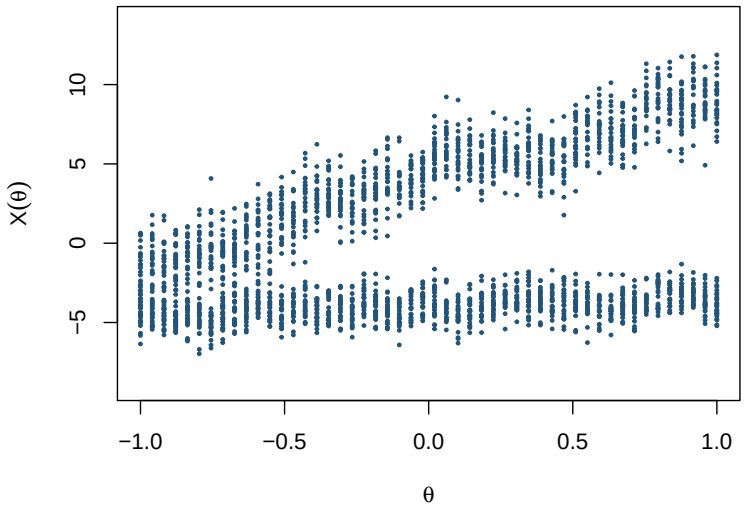

Figure 7.2: Data obtained from a two-component GP mixture.

MCMC, after 1K burn-in removed, we retain every 20^{th} iterate to obtain 1K samples from the posterior. Figure 7.3 shows the marginal posterior densities for both the number of clusters and the concentration parameter α. We can see the largest posterior probability is for two clusters with a non-negligible probability of three clusters being necessary to fit the data. Given that the simulated data is from a two-component mixture, this is expected.

Since we wish to use this model for emulation, the predictive distribution is of primary interest and so we can predict from this by averaging over the posterior samples (obtained with MCMC) as described in Section 7.5. Using a grid of new points in θ-space we obtain Figure 7.4 where we see both component mean functions are adequately captured. Additionally, the fitted model is able to account for the different variances used to generate the data. We can also assess the fit by looking at the univariate predictive densities by choosing points in θ-space. Here we choose points where we had training data so we can compare the predictive density to these. Figure 7.5 shows these univariate densities at three training points, where we can see the bimodal behaviour sufficiently accounted for.

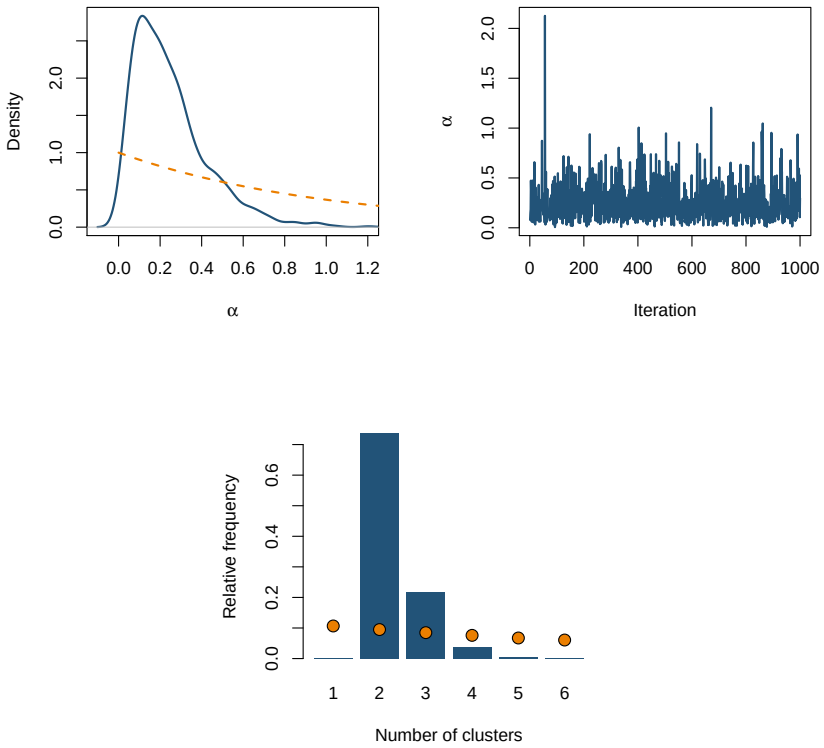

Figure 7.3: Marginal posterior density for α (top left) with prior density overlaid in orange, trace plot for α (top right) and posterior probabilities for the number of clusters (bottom) with prior probabilities overlaid in orange.

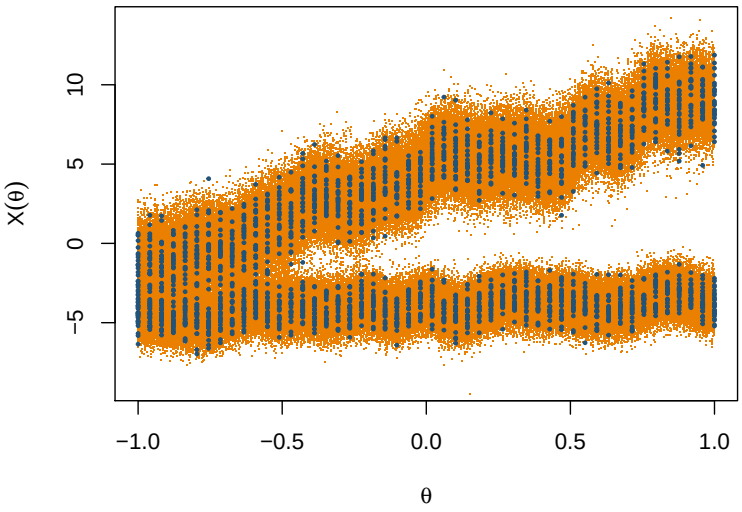

Figure 7.4: Samples from the posterior predictive distribution (orange) with training data overlaid (blue).

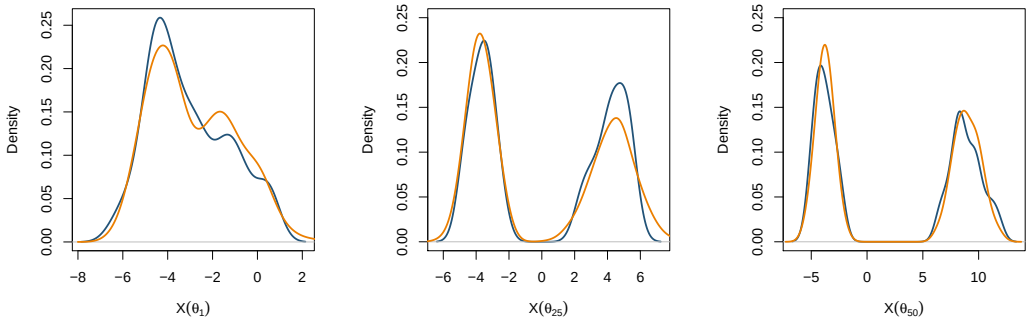

Figure 7.5: Univariate predictive densities (orange) with density of training data (blue).

Chapter 8

Calibration

8.1 Introduction

The aim of this project is to find a flexible and efficient approach for inferring the rate constants of stochastic kinetic models as defined in Chapter 2. In Chapter 7 we constructed a flexible model that used Dirichlet processes (Chapter 6) to allow for the number of clusters to vary. To allow for spatial variation we use Gaussian process regression (Chapter 4) which will allow us to interpolate the distribution between training inputs.

In this chapter we investigate the use of a Dirichlet process mixture of Gaussian processes as an emulator for stochastic kinetic model output. We train the emulator by fitting the model to training data. Using the techniques described in Chapter 3, we will use the emulator as part of a larger inference scheme that will sample from the posterior distribution of the stochastic rate constants, given data at discrete time points that may be incomplete and subject to measurement error.

8.2 Schlögl system

We will infer the stochastic kinetic rate constants, $c_1, \ldots, c_4$, or rather, the log of these constants, $\boldsymbol{\theta} = \log \boldsymbol{c}$, using data from the Schlögl system (Section 2.7) with reaction

equations

$$
\begin{aligned}
R_1: &\quad 2X_1 + X_2 \xrightarrow{e^{\theta_1}} 3X_1, \\
R_2: &\quad 3X_1 \xrightarrow{e^{\theta_2}} 2X_1 + X_2, \\
R_3: &\quad X_3 \xrightarrow{e^{\theta_3}} X_1, \\
R_4: &\quad X_1 \xrightarrow{e^{\theta_4}} X_3.
\end{aligned}
$$

8.2.1 Data

As described in Chapter 3, output from SKMs is observed discretely in time with some observation noise. From these observations we wish to infer the parameters of the underlying model. To provide a challenging data-poor scenario, we will start our analysis by only looking at noisy data of the second specie X_2 and so we will only emulate the system data for this one specie. The following noisy X_2 data, Figure 8.1, are drawn from the Schlögl system where we assume that the variance of the noise is known to be σ^2 with $\sigma = 1000$ at times $t = 1, 2, \ldots, 20$. That is,

$$y_t(\boldsymbol{\theta}) = x_t(\boldsymbol{\theta}) + \epsilon_t, \quad t = 1, 2, \ldots, 20,$$

where the ϵ_t are i.i.d with $\epsilon_t \sim N(0, 1000^2)$. Each point will be observed from a different experiment, for example in a real world scenario you could imagine 20 petri dishes, each initialised with the same initial conditions, and then at time $t = 1$ we record the measurement in the first petri dish, at $t = 2$ we record the measurement in the next petri dish and so on until $t = 20$. The data in Figure 8.1 are simulated with the parameter choice $\boldsymbol{\theta} = \log(3 \times 10^{-7}, 10^{-4}, 0.000773, 3.276)$ (Owen et al., 2015) that ensures we have biomodal output to provide a challenging emulation scenario.

The first step in the calibration is to fit the emulator to training data. The approach we take is to fit independent emulators at each time point. At each of these time points, we will fit the emulators to training data in a Latin hypercube design where the same design will be used at each time point. Having $T = 20$ independent emulators is computationally beneficial as it means that each can be fit in parallel.

Once we have the emulator trained at each time point, we will then have a mechanism for approximately simulating from the model, that is $\hat{\boldsymbol{X}}_t \sim \hat{\pi}(\boldsymbol{x}_t|\boldsymbol{\theta})$, for different values of the rate constants. Of course we could use Gillespie's direct method, but we anticipate that use of the emulator will result in substantial efficiency gains. We assess accuracy by comparing the posteriors obtained under the emulation approach to those obtained using the Markov jump process (MJP) as the inferential model.

We apply the SMC scheme as described in Section 3.4.3, where the distribution at a new proposed value $\boldsymbol{\theta}^* = \log \boldsymbol{c}^*$ will be approximated by sampling from the predictive distribution for the independent Dirichlet Process mixtures of Gaussian processes fitted to training data at each time $t = 1, 2, \ldots, 20$. In what follows, we provide implementation details.

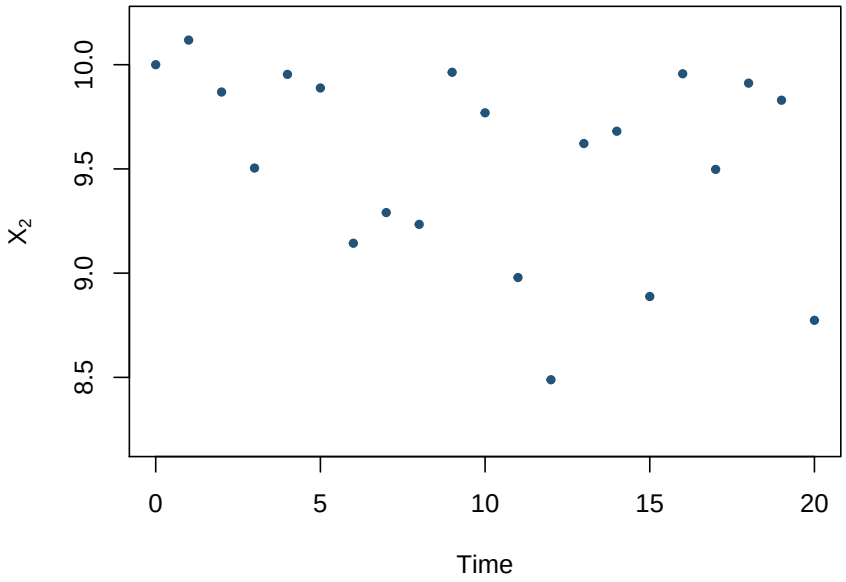

Figure 8.1: Noisy observations from the Schlögl system at 20 time points. Note that the output has been rescaled via $X_2/10000$.

8.2.2 Training data

To generate training data, we require N points in $\boldsymbol{\theta}$ space that will each have M replicate observations of SKM output. At each of these N points in a Latin hypercube design (LHD), Section 5.2, in $\boldsymbol{\theta}$ space, we will simulate from the model M times in order to obtain M replications with the aim of fitting a Dirichlet Process mixture of Gaussian Processes (DPMGP) at each time point independently.

The following analysis uses $N = 200$ points and $M = \{10, 20, 40\}$ replicates of training data at each time point. We found that the choice of N gave an adequate balance between accuracy and efficiency and investigate the effect of increasing M. The initial counts of the species are $x_1(0) = 250, x_2(0) = 10^5, x_3(0) = 2 \times 10^5$ which is the typical choice used to observe the bifurcating behaviour (Owen et al., 2015). To ensure that the parameters are on a reasonable scale, we rescale the data by letting $X_2 = X_2/10000$ and $\phi_i = \frac{\theta_i - a_i}{b_i - a_i}$ where a_i and b_i are the lower and upper bounds of the LHD. Here we choose the bounds

such that they are in the tails of the posterior distribution found in Owen et al. (2015). The bounds are chosen to be $-15.6 < \theta_1 < -14.6, -10 < \theta_2 < -8.5, -8 < \theta_3 < -4.5$ and $1 < \theta_4 < 2$. If previous analysis was not available to us we could choose these bounds to be in the tails of the prior distributions for $\boldsymbol{\theta}$. The training inputs can be seen in Figure 8.2 where the Latin hypercube was constructed using the maximin method as described in Section 5.2.

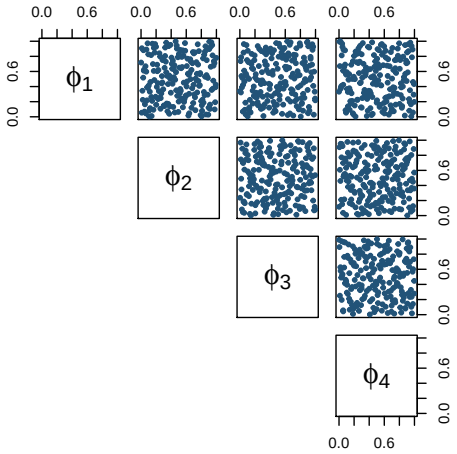

Figure 8.2: Lating hypercube design of training points with $N = 200$.

The parameter choices for the prior distributions are $V_0 = 10 \times I_p, a_\sigma = 1, b_\sigma = 0.01, \boldsymbol{a_r} = (0, 0, 0, 0)^\top, b_{\boldsymbol{r}} = 0.1 \times I_d, a_\nu = 1, b_\nu = 0.01, a_\alpha = 1, b_\alpha = 1$. The prior mean, $\boldsymbol{m}_0$, for the coefficients of the mean function $m(\boldsymbol{\theta}) = \beta_0 + \sum_{i=1}^{p} \beta_i \theta_i$ are chosen to be the least squares estimates of the coefficients found when fitting a regression to the training data. Figure 8.3 shows the prior placed on α and the induced prior on k, the number of clusters. Figure 8.4 shows the prior densities for the remaining hyperparameters.

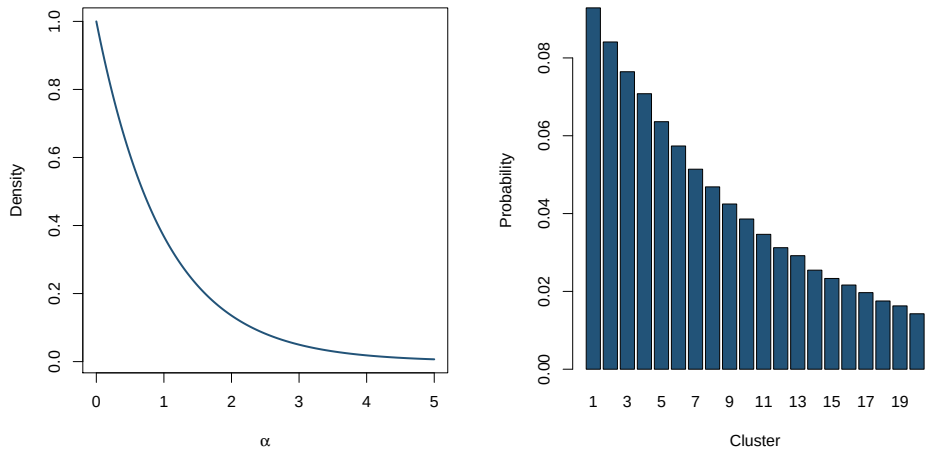

Figure 8.3: Prior density of α (left) and induced prior on the number of clusters (right).

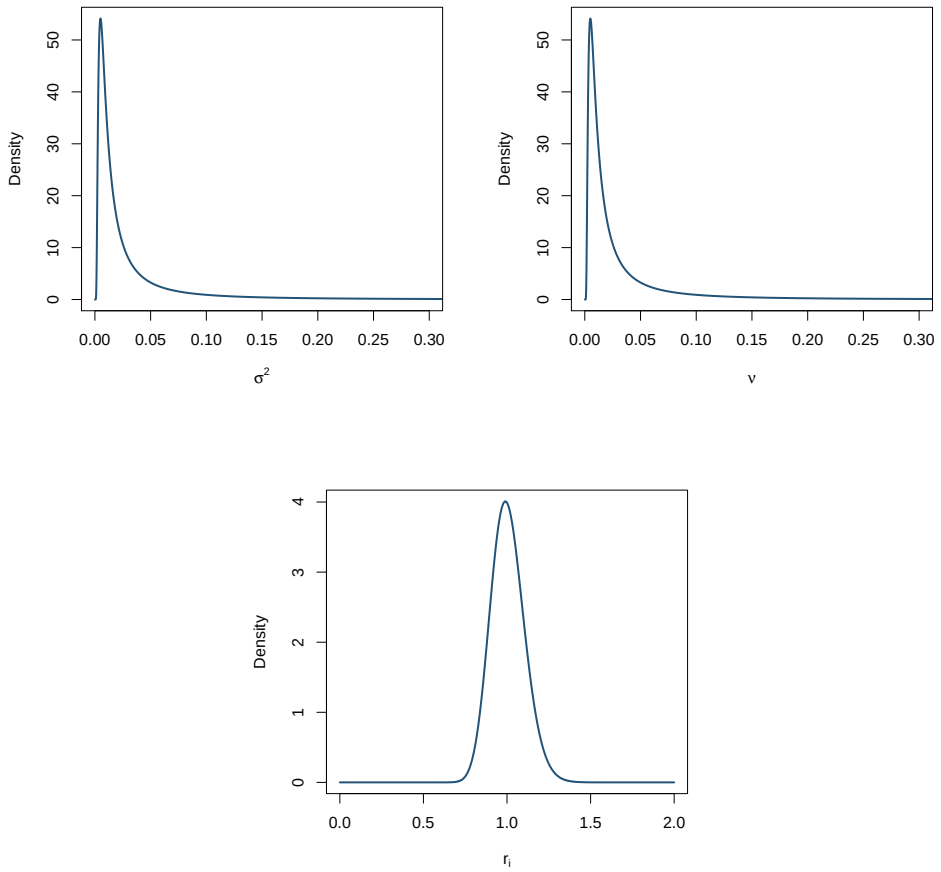

Figure 8.4: Prior density of σ^2 (left), ν (middle) and r_i (right).

We fit independent emulators at each time point by inferring the hyperparameters. This required running the MCMC scheme described in Chapter 7 in parallel for the training data at each time point. This meant that we could fit at each time independently. After a 2K iteration burn-in was removed, 10K samples were thinned by 10 to give 1K samples from the hyper-parameter posterior at each time point. Trace plots A.3.4 were examined in order to determine that the chain was mixing well between the number of clusters. The total run times for fitting are given towards the end of the chapter, where the algorithms were coded in C and ran on a high performance cluster with multiple cores (Intel Xeon 2.2 GHz processors on 20 cores).

8.2.3 Inference

We will use the sequential Monte Carlo scheme as described in Section 3.4.3, to sample from the marginal posterior distributions under the MJP and emulation approach. The observations $\boldsymbol{y}$ are assumed to be noisy measurements of the true system data $\boldsymbol{x}$ with known noise σ^2. To provide a challenging inference scenario, we took $\sigma = 1000$, which is an order of magnitude smaller than typical output values. This translates to a known error of $\sigma = 0.1$ when in the rescaled X_2 space. We could infer the observational error from the data as part of the inference scheme but keep this fixed to keep focus on inferring the stochastic rate constants.

We place truncated Gamma priors on ϕ_i where we truncate at 1. We choose two parameters α_ϕ, λ_ϕ for the prior $\phi_i \sim Ga(\alpha_\phi, \lambda_\phi)|_{\phi<b_i}$ such that the prior is not too informative. Specifically, we place independent $Ga(2, 5)$ priors on each ϕ_i as shown in Figure 8.5 with the ground truth values of the ϕ_i.

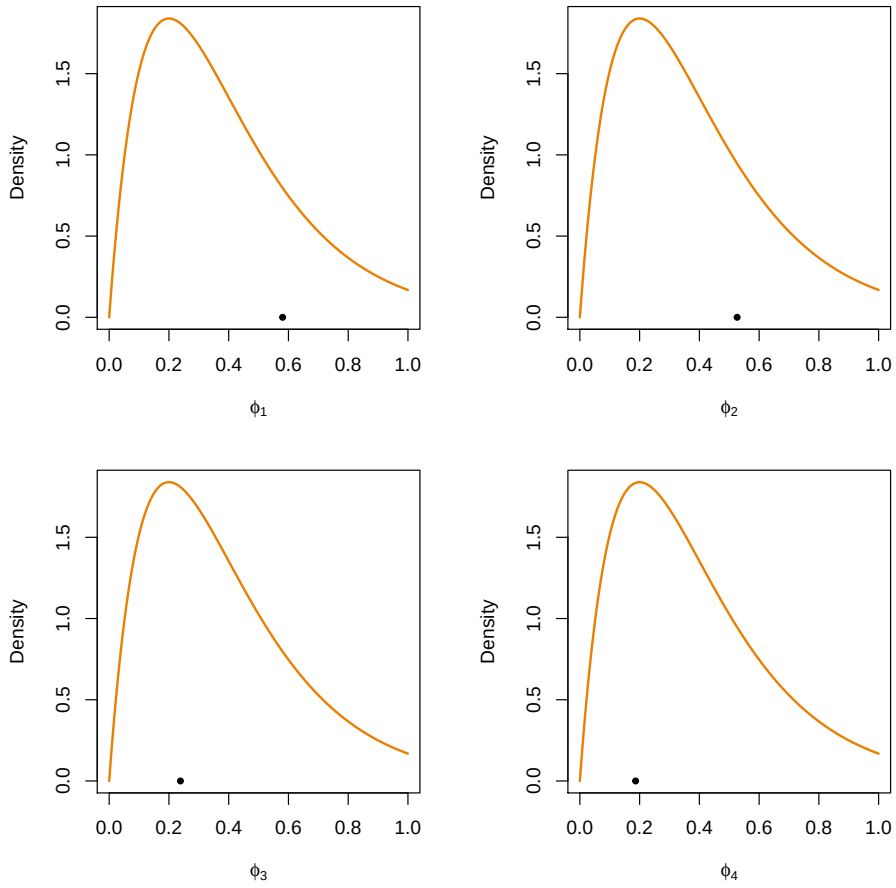

Figure 8.5: Prior density of ϕ_i with true value (in ϕ space) in black.

The SMC scheme for inference under the MJP is given by Algorithm 5. For completeness, we additionally provide the SMC scheme that we use for calibration via use of the emulator in Algorithm 9.

Note that to alleviate particle degeneracy, we include an additional rejuvenation step. Essentially, if the number of distinct particles is less than $N_p/2$, we draw new parameter and state values from a Metropolis-Hastings kernel that has the target posterior as its invariant distribution. See Cappé et al. (2007) for further details. For the proposal mechanism, we use a normal random walk on the logit scale, with innovation variance calculated from the particle set. Including this step in Algorithm 5 is straightforward and can be achieved by simply replacing draws from the MJP (via Gillespie's direct method) with draws from the DPMGP (emulator).

We now compare and contrast calibration results under the MJP and emulation approach.

Algorithm 9 Bootstrap filter with emulation

- Initialise
  At time $t = 0$, for $i = 1, \ldots, P$
  1. Sample $\boldsymbol{\theta}^{(i)} \sim \pi(\boldsymbol{\theta})$.
  2. Sample $\tilde{\boldsymbol{X}}_{t+1}^{(i)} \sim \hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{\theta}^{(i)})$ using the DPMGP emulator.
  3. Calculate and assign a weight to each particle $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)})$ where $\tilde{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma)$.
  4. Resample $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)})$ N_p times according to the normalised weights $\{w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N_p)}\}$.
- For times $t = 1, \ldots, T$ and $i = 1, \ldots, N_p$
  1. Sample $\tilde{\boldsymbol{X}}_{t+1}^{(i)} \sim \hat{\pi}(\boldsymbol{x}_{t+1}|\boldsymbol{\theta}^{(i)})$ using the DPMGP emulator.
  2. Calculate and assign a weight to each particle $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)})$ where $\tilde{w}_{t+1}^{(i)} = \pi(\boldsymbol{y}_{t+1}|\tilde{\boldsymbol{x}}_{t+1}^{(i)}, \Sigma)$.
  3. Resample $(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{x}}_{t+1}^{(i)})$ N_p times according to the normalised weights $\{w_{t+1}^{(1)}, \ldots, w_{t+1}^{(N_p)}\}$.
  4. If the number of distinct $\boldsymbol{\theta}$ samples is less than $N_p/2$, rejuvenate:
     Propose $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i)})$ and $\boldsymbol{X}_s^* \sim \hat{\pi}(\boldsymbol{x}_s|\boldsymbol{\theta}^*)$, $s = 1, \ldots, t+1$. Accept the pair $(\boldsymbol{\theta}^*, \tilde{\boldsymbol{x}}_{t+1}^*)$ with probability $\min(1, A)$ where

$$A = \frac{\pi(\boldsymbol{\theta}^*)}{\pi(\boldsymbol{\theta}^{(i)})} \times \frac{\prod_{s=1}^{t+1} \pi(\boldsymbol{y}_s|\boldsymbol{x}_s^*, \Sigma)}{\prod_{s=1}^{t+1} \pi(\boldsymbol{y}_s|\boldsymbol{x}_s^{(i)}, \Sigma)} \times \frac{q(\boldsymbol{\theta}^{(i)}|\boldsymbol{\theta}^*)}{q(\boldsymbol{\theta}^*|\boldsymbol{\theta}^{(i)})}$$

MJP as inferential model

We run the SMC scheme with Gillespie's direct method as the simulator. The posterior obtained using this method will be considered as the true posterior, as we are not using any type of emulator or approximation.

We run the SMC scheme with 500K particles to obtain a good approximation to the posterior without too much computational effort. The run time for this took just under 63 hours. Results are shown in Figure 8.6.

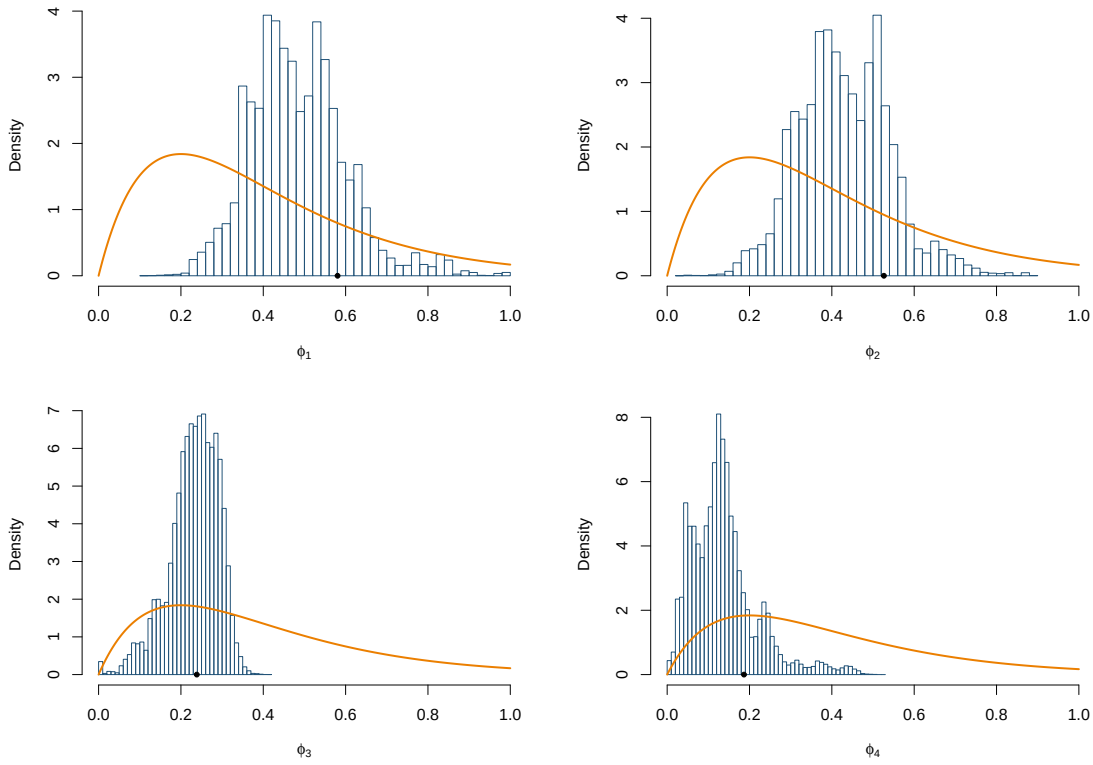

Figure 8.6: Histogram of posterior samples for each ϕ_i (at time $t = 20$) with prior (orange) overlaid and 'true' point (black).

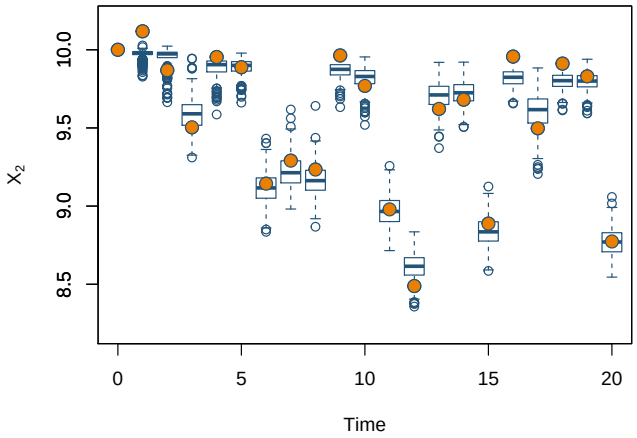

Figure 8.7: Boxplot of samples of $x_t|y$ against t, with observations overlaid (orange).

The true values of the rate constants used to simulate the noisy data can be found well within the bulk of the marginal posterior distributions. We can see that the analysis has been informative, with a reduction in uncertainty from prior to posterior. Figure 8.7 shows boxplots of the marginal posteriors of each latent state $\boldsymbol{x}_t$. As expected, the simulated data are consistent with the posteriors.

Emulator as inferential model

We now investigate the posterior obtained using the DPMGP emulator. Recall that the posterior here is an approximation of the true posterior since we are no longer simulating actual system data but instead simulating from the emulator within the inference scheme. The interpolation between the training points will therefore introduce some additional uncertainty into the posterior distribution. This problem can be alleviated by increasing the number of training points, either by increasing the number of points in the LHD (N) or increasing the number of replicates at each point in $\boldsymbol{\theta}$ space (M). Due to the block structure of the covariance matrices as described earlier, it is preferable to increase M rather than N as we only require calculating the inverse and determinant of $N \times N$ matrices regardless of the size of M. We will therefore investigate the effect the number of replicates has on the accuracy of the posterior distributions obtained.

$M = 10$ replicates

We begin by fitting the DPMGP emulators at each time point with just $M = 10$ replicates at each of the training inputs.

Figure 8.8 shows histograms of each marginal parameter posterior. Although it is clear that the posterior samples are consistent with the ground truth, they are generally inconsistent with the posterior under the MJP (see Figure 8.6). The observed multi-modality could be due to particle degeneracy, since output of the DPMGP emulator with $M = 10$ replicates naturally exhibit a higher variance than that of an exact simulator. Consequently, draws from the emulator are less likely to be 'consistent' with the data (given the measurement error variance) resulting in relatively few $\boldsymbol{\theta}$ samples with reasonable weight. This additional variance can be seen in Figure 8.9.

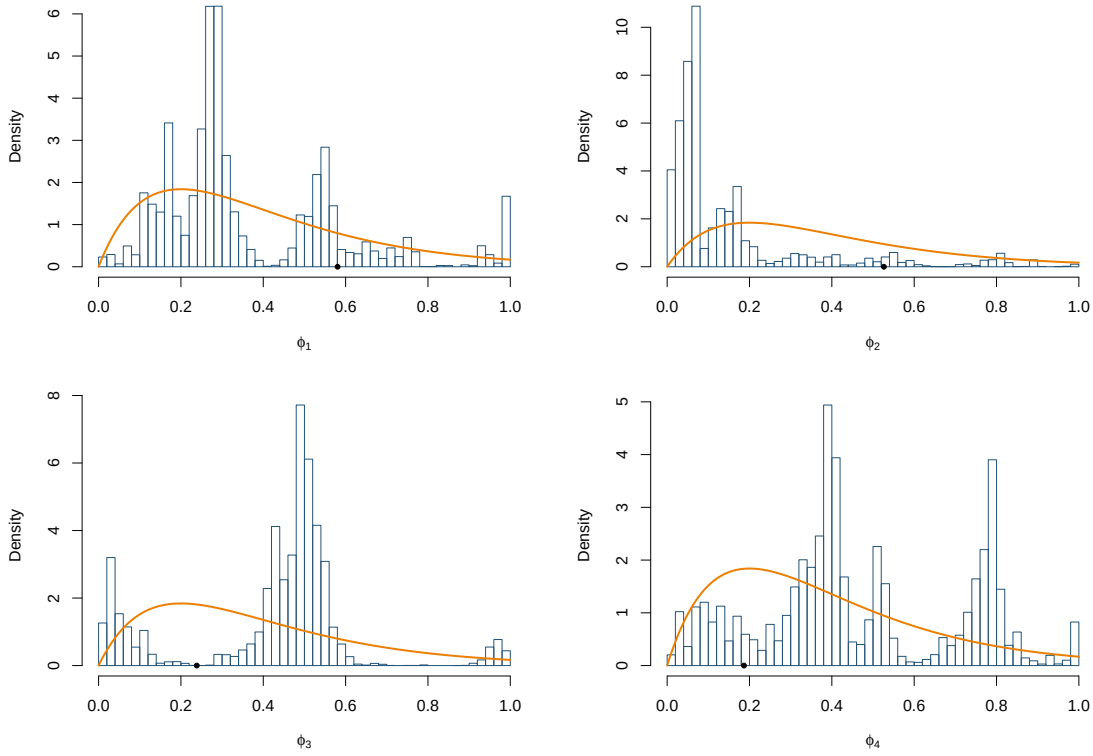

Figure 8.8: Histogram of posterior samples for each ϕ_i (at time $t = 20$) with prior (orange) overlaid and 'true' point (black).

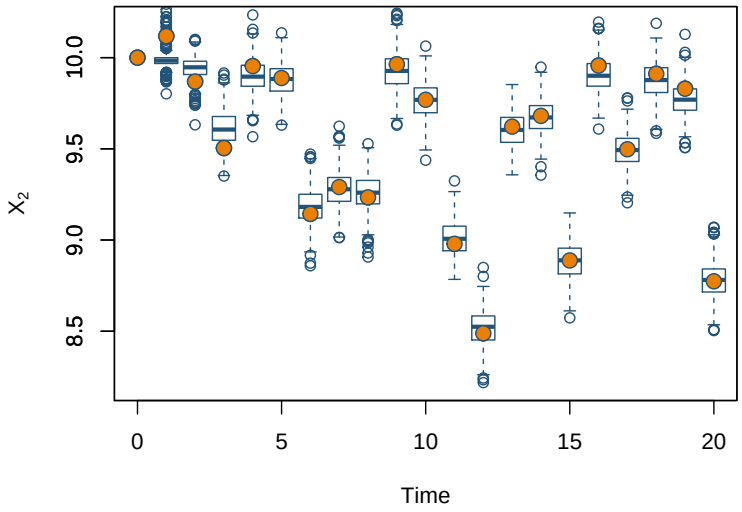

Figure 8.9: Boxplot of samples of $x_t|y$ against t, with observations overlaid (orange).

$M = 20$ replicates

Here we fit the emulators to training data at the same LHD of $N = 200$ points but now with $M = 20$ replicates at each training point, double that of the previous case.

Figure 8.10 shows the marginal posteriors obtained using these independent emulators at each time point. Posterior variance is reduced (relative to the $M = 10$ case) and the majority of the sampled values of ϕ_1 and ϕ_3 are consistent with the true posteriors shown in Figure 8.6. Nevertheless, there is still some considerable mismatch between the marginal posteriors for ϕ_2 and ϕ_4 under the DPMGP ($M = 20$) and those under the MJP. Figure 8.11 shows boxplots of samples from the marginal posterior of each latent state. Again, we see greater variability compared to Figure 8.7.

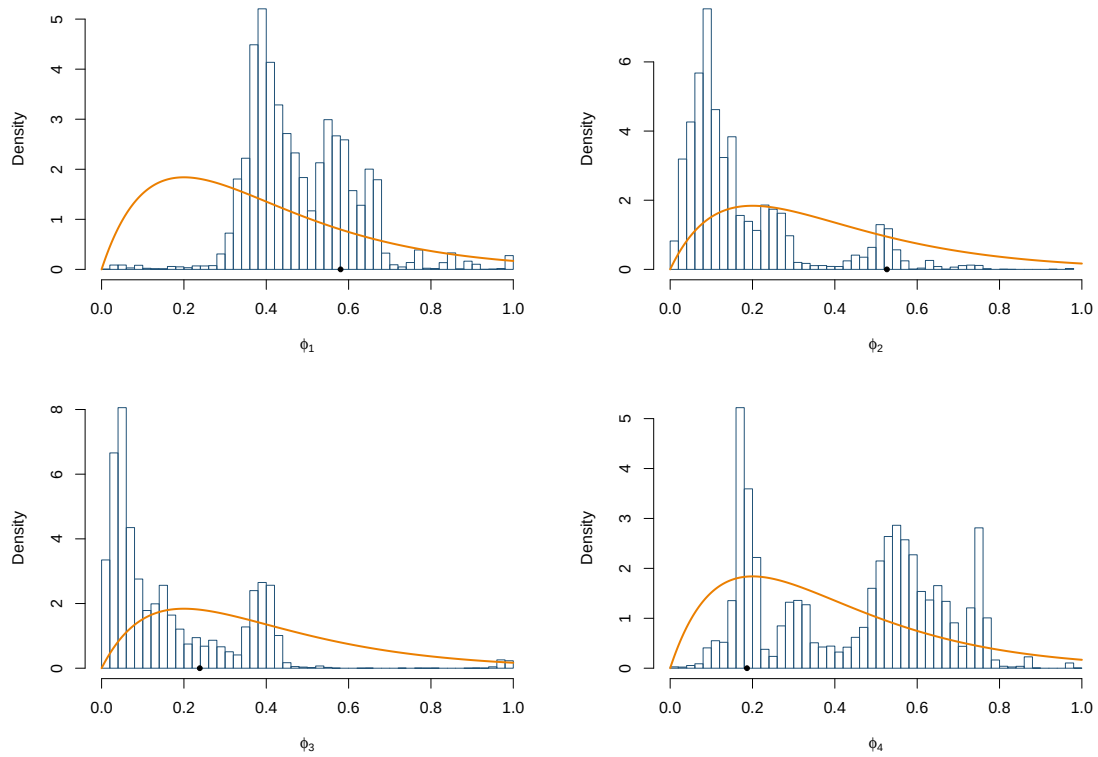

Figure 8.10: Histogram of posterior samples for each ϕ_i (at time $t = 20$) with prior (orange) overlaid and 'true' point (black).

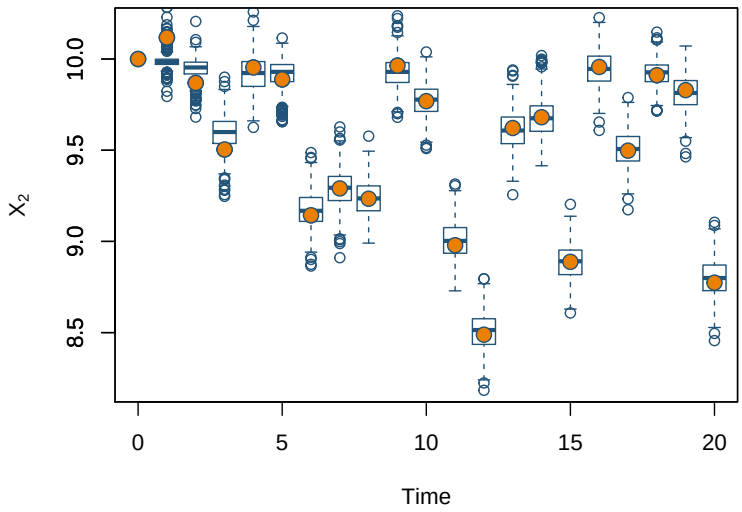

Figure 8.11: Boxplot of samples of $x_t|y$ against t, with observations overlaid (orange).

$M = 40$ **replicates**

In this final case we double the number of replicate observations again at each point in ϕ space. Clearly as we increase the number of replicates, the number of total training data observations increases, therefore increasing the fitting time in each case. So although our emulator will become more accurate, the time taken to fit each will increase. Computational considerations are discussed below.

The posterior densities for ϕ_3, ϕ_4 seem to be consistent with the true values used to simulate the noisy data. Moreover, compared to the $M = 10$ and $M = 20$ cases, there is less mismatch between posteriors obtained under the DPMGP and those under the MJP.

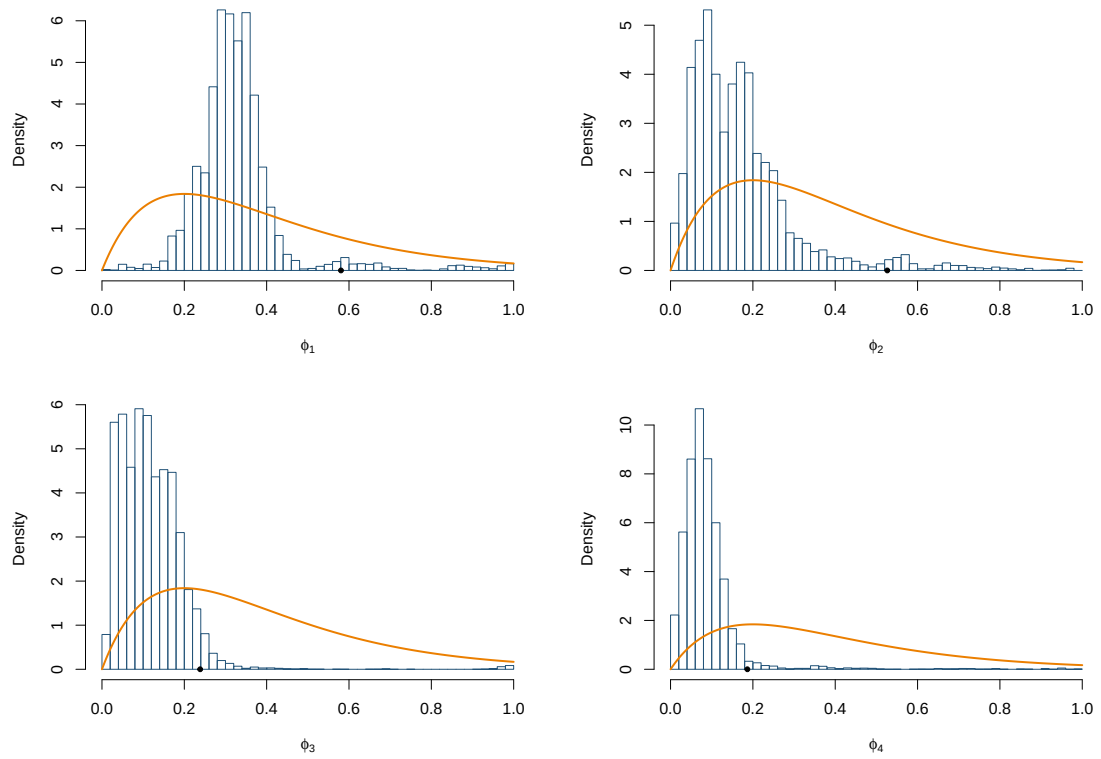

Figure 8.12: Histogram of posterior samples for each ϕ_i (at time $t = 20$) with prior (orange) overlaid and 'true' point (black).

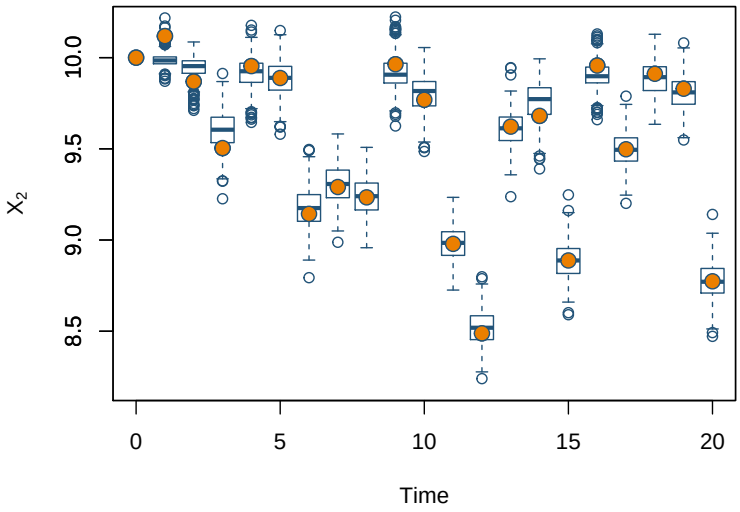

Figure 8.13: Boxplot of system data $\boldsymbol{x}$ particles at each time point with observation overlaid (orange).

Comparisons and computational cost

The run time for the inference here was 15 hours, much faster than using Gillespie, however, the fitting time of the emulator of course needs to be accounted for, and this takes the total time to around 6 days. Table 8.2 gives marginal posteriors means under each scheme as well as standard deviations in parentheses.

Model	Run time (hours)	Training time (hours)
MJP	63	—
DPMGP ($M = 10$)	14	46
DPMGP ($M = 20$)	15	96
DPMGP ($M = 40$)	15	145

Table 8.1: Computation cost for each model.

Model	ϕ_1	ϕ_2	ϕ_3	ϕ_4
MJP	0.503 (0.116)	0.451 (0.108)	0.226 (0.058)	0.144 (0.085)
DPMGP ($M = 10$)	0.388 (0.194)	0.134 (0.167)	0.401 (0.184)	0.470 (0.238)
DPMGP ($M = 20$)	0.522 (0.134)	0.203 (0.162)	0.197 (0.180)	0.477 (0.218)
DPMGP ($M = 40$)	0.338 (0.112)	0.181 (0.132)	0.127 (0.074)	0.088 (0.079)

Table 8.2: Table of marginal posterior means with standard deviations in parentheses.

Chapter 9

A variational approach

9.1 Introduction

The aim of this thesis is to find a fast but accurate approximation to SKM output that can then be used in an inference scheme to infer the stochastic rate parameters. In the previous chapter we used a DPMGP model as an emulator. Although this model was flexible enough to model a system such as the Schlögl system, the computational burden associated with fitting the emulator to training data precluded its use as a viable alternative to the MJP as an inferential model.

In this chapter we introduce the variational approximation method before looking at applying the variational approach to our problem of fitting the DPMGP emulator to training data. This means we will be introducing a further approximation into the analysis, but this trade-off will give a reduction in the total computational cost of using the emulator.

9.2 Variational inference

Previously we employed MCMC methods to fit the emulator by sampling from the posterior distribution of the DPMGP parameters. Although MCMC allowed us to draw from such a complex posterior, it can suffer from convergence issues as well as lack efficiency. We now outline an approximate, computationally cheap alternative to using MCMC – the variational approach.

Developed in the machine learning field, variational inference is used to approximate complex posterior distributions through use of simpler distributions. In what follows, we give a brief overview of variational inference but refer the reader to Blei et al. (2017) and Blei et al. (2006) for more detailed discussion.

Let $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ be the set of observed data and $\boldsymbol{z} = (z_1, \ldots, z_m)^T$ be the set of latent variables with joint density $\pi(\boldsymbol{z}, \boldsymbol{x})$. Performing inference to learn about the latent variables given the data here means computing the posterior density $\pi(\boldsymbol{z}|\boldsymbol{x})$. Bayes' theorem gives

$$\pi(\boldsymbol{z}|\boldsymbol{x}) = \frac{\pi(\boldsymbol{z}, \boldsymbol{x})}{\pi(\boldsymbol{x})},$$

and recall that the denominator is the marginal density of the data, often called the evidence, and is calculated by marginalising out the latent variables from the joint density,

$$\pi(\boldsymbol{x}) = \int \pi(\boldsymbol{z}, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{z}.$$

When performing variational inference, a family of densities, $\mathcal{Q}$, over the latent variables is specified where $q(\boldsymbol{z}|\nu) \in \mathcal{Q}$ is an approximation to the posterior (with dependence on variational parameters ν which we suppress from now for ease of notation). The aim is to then find the best approximation which is the one closest to $\pi(\boldsymbol{z}|\boldsymbol{x})$ in Kullback-Leibler (KL) divergence. The KL divergence (Kullback and Leibler, 1951) is a measure of proximity between two densities f and g computed by taking the expectation (with respect to f) of the logarithmic difference between the two, that is

$$\mathrm{KL}(f(x)||g(x)) = \int_{-\infty}^{\infty} f(x) \log \frac{f(x)}{g(x)} \, \mathrm{d}x.$$

This measure is asymmetric, hence $\mathrm{KL}(f(x)||g(x)) \neq \mathrm{KL}(g(x)||f(x))$, and is minimised when the two densities are exact, that is, $f(\cdot) = g(\cdot)$. In variational inference, the following optimisation is performed

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z})}{\arg\min} \, \mathrm{KL}(q(\boldsymbol{z})||\pi(\boldsymbol{z}|\boldsymbol{x})) \tag{9.1}$$

as this will give us $q^*(\boldsymbol{z})$ that is closest to the posterior density $\pi(\boldsymbol{z}|\boldsymbol{x})$.

Looking again at the KL divergence and expanding the conditional density $\pi(\boldsymbol{z}|\boldsymbol{x})$ we have

$$\begin{aligned} \mathrm{KL}(q(\boldsymbol{z})||\pi(\boldsymbol{z}|\boldsymbol{x})) &= \mathrm{E}_q[\log q(\boldsymbol{z})] - \mathrm{E}_q[\log \pi(\boldsymbol{z}|\boldsymbol{x})] \\ &= \mathrm{E}_q[\log q(\boldsymbol{z})] - \mathrm{E}_q[\log \pi(\boldsymbol{z}, \boldsymbol{x})] + \log \pi(\boldsymbol{x}). \end{aligned} \tag{9.2}$$

Hence, computing the KL divergence requires computing the log evidence, $\log \pi(\boldsymbol{x})$.

Given that the log evidence is hard to compute, we instead optimise the evidence lower bound (ELBO)

$$\mathrm{ELBO}(q) = \mathrm{E}_q[\log \pi(\boldsymbol{z}, \boldsymbol{x})] - \mathrm{E}_q[\log q(\boldsymbol{z})]. \tag{9.3}$$

We now show that the ELBO in (9.3) is equivalent to the KL divergence in (9.2) up to an additive constant. We have that

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{z})||\pi(\boldsymbol{z}|\boldsymbol{x})) &= \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{q(\boldsymbol{z})}{\pi(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\
&= -\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{\pi(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})} d\boldsymbol{z} \\
&= -\left(\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{\pi(\boldsymbol{z}, \boldsymbol{x})}{q(\boldsymbol{z})} d\boldsymbol{z} - \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \pi(\boldsymbol{x}) d\boldsymbol{z} \right) \\
&= -\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{\pi(\boldsymbol{z}, \boldsymbol{x})}{q(\boldsymbol{z})} d\boldsymbol{z} + \log \pi(\boldsymbol{x}) \int_{\boldsymbol{z}} q(\boldsymbol{z}) d\boldsymbol{z} \\
&= -\int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{\pi(\boldsymbol{z}, \boldsymbol{x})}{q(\boldsymbol{z})} d\boldsymbol{z} + \log \pi(\boldsymbol{x}) \\
&= -\mathrm{ELBO}(q) + \log \pi(\boldsymbol{x}). \qquad (9.4)
\end{aligned}$$

Equation 9.4 shows that the ELBO is simply the negative KL divergence plus the log evidence, which is a constant with respect to $q(\boldsymbol{z})$, thus, maximising the ELBO is equivalent to minimising the KL divergence and so this becomes the objective function that requires optimisation. It remains that we can specify a variational family $\mathcal{Q}$ to fully describe the optimisation problem.

One of the simplest families to use is the mean-field variational family which assumes that each of the latent variables are mutually independent. Each latent variable has its own density such that

$$q(\boldsymbol{z}) = \prod_{j=1}^{m} q_j(z_j).$$

Each of these densities $q_j(z_j)$ is found through optimisation by maximising the ELBO in 9.3.

If we now focus on the j^{th} latent variable z_j, the variational distribution can be dervied by considering the conditional distribution given all of the other parameters, a similar idea to Gibbs sampling in Section 3.3.1. By fixing all of the other variational factors $q_l(z_l)$ for $l \neq j$, finding the optimal $q_j(z_j)$ is an optimisation problem which can be solved through coordinate ascent variational inference (Bishop, 2006). By fixing all other variational factors, considering the ELBO as a function of $q_j(z_j)$ and differentiating, the optimal density is found to be proportional to the exponential of the expected log conditional density. That is,

$$q_j^*(z_j) \propto \exp\{\mathrm{E}_{-j}[\log \pi(z_j|\boldsymbol{z}_{-j}, \boldsymbol{x})]\}.$$

9.3 Nonparametric mixture of Gaussian processes

The largest portion of computational cost of the DPMGP emulator method was the fitting of the emulator to training data at each time point. Although each time could be fit in parallel, every single training point (out of a total of $N \times M$) had to be allocated given all of the other allocations, meaning this step could not be parallelised and thus caused the fitting to be computationally prohibitive. We therefore turn to a variational approach to fitting a mixture of GPs, specifically the model described in Ross and Dy (2013). This model uses a hierarchical setup for the latent functions and so instead of the hyperparemeters of the GPs being drawn from the DP (as in the DPMGP model earlier) the latent functions themselves are drawn from the DP.

9.3.1 Model overview

To describe the model we begin by assuming that there is just one replicate at each point in input space. Let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)^T$ be the N-vector of inputs and $\boldsymbol{X} = (x_1, \ldots, x_N)^T$ be the N-vector of observations and σ_e^2 be the observational error. The covariance function is the squared exponential with elements $(K)_{ij} = \sigma^2 \exp\{-(\theta_i - \theta_j)^2/2l^2\}$. Below we give the model from Ross and Dy (2013). The authors also have certain constraints on the data which they use as a mechanism for incorporating expert opinion, something which we omit. The variational approximation requires us to use the stick-breaking representation of the Dirichlet process, as discussed in Section 6.3.1, where the finite approximation requires setting a truncation parameter L. For $k = 1, \ldots, L$,

$$
\begin{aligned}
\boldsymbol{f}^{(k)}|\boldsymbol{\theta} &\sim N(\mathbf{0}, K^{(k)}) \\
x_i|\boldsymbol{f}, \boldsymbol{z}_i, \sigma_e^2 &\sim N(\boldsymbol{z}_i^\top f_i, \sigma_e^2) \\
\boldsymbol{z}_i|\boldsymbol{\rho}_i &\sim Cat(\boldsymbol{\rho}_i) \\
\rho_{ik}|\boldsymbol{v} &= v_k \prod_{j=1}^{k-1}(1 - v_j) \\
v_k|\alpha &\sim Beta(1, \alpha).
\end{aligned}
$$

As can be seen, this differs slightly from the earlier DPMGP model. Firstly, the actual GP functions are now drawn from the DP, as opposed to the GP hyperparameters being the variables drawn from the DP. This allows for the observations to be distributed around these GP functions with error σ_e^2. Further, allocation of each observation x_i is through a binary indicator vector $\boldsymbol{z}_i$. This change allows for a variational distribution to be found as the likelihood of the observations can be written as a simple product of normals over the clusters and number of observations, something which could not be done for the

previous DPMGP model. We now also assume a zero mean function and a slightly altered covariance function for simplicity.

The standard variational approach as discussed above is to use the mean-field variational family and so we assume the following factorisation of the joint distribution as

$$q^*(\{\boldsymbol{f}^{(L)}\})q^*(\boldsymbol{v})q^*(\boldsymbol{Z}).$$

The following expressions are the variational distributions as found in Ross and Dy (2013). The variational distribution over the latent functions is

$$q^*(\{\boldsymbol{f}^{(L)}\}) = \prod_{k=1}^{L} N(\boldsymbol{\mu}^{(k)}, C^{(k)})$$

where

$$\begin{aligned} C^{(k)} &= (K^{(k)^{-1}} + R^{(k)})^{-1} \\ \boldsymbol{\mu}^{(k)} &= C^{(k)} R^{(k)} \boldsymbol{x} \end{aligned}$$

and

$$R^{(k)} = \frac{1}{\sigma_e^2} \text{diag}(\text{E}[\boldsymbol{Z}]_{1k}, \ldots, \text{E}[\boldsymbol{Z}]_{Nk}).$$

The matrix $R^{(k)}$ is simply a diagonal matrix with diagonal element i given by the probability of observation i being in cluster k. The variational distribution over $\boldsymbol{v}$, the variables used to construct the stick-lengths in the stick-breaking representation of the DP, is

$$q^*(\boldsymbol{v}) = \prod_{k=1}^{L} Beta\left(1 + \sum_{i=1}^{N} \text{E}[\boldsymbol{Z}]_{ik}, \alpha + \sum_{j=k+1}^{L} \sum_{i=1}^{N} \text{E}[\boldsymbol{Z}]_{ik}\right).$$

The variational distribution over $\boldsymbol{Z}$, the indicator variables of the observations, is

$$q^*(\boldsymbol{Z}) = \prod_{i=1}^{N} \prod_{k=1}^{L} r_{ik}^{z_{ik}}$$

where

$$r_{ik} = \frac{\rho_{ik}}{\sum_{k=1}^{L} \rho_{ik}}$$

and

$$\log \rho_{ik} = \log \frac{1}{\sqrt{2\pi\sigma_e^2}} - \frac{1}{2\sigma_e^2}\left(x_i^2 - 2x_i \text{E}\left[f_i^{(k)}\right] + \text{E}\left[f_i^{(k)^2}\right]\right) + \text{E}[\log v_k] + \sum_{j=1}^{k-1} \text{E}[\log(1 - v_j)].$$

For large sets of training data some of these calculations may become infeasible, in particular the covariance matrix will become rather large. As a consequence of the replicate observations at each point in $\boldsymbol{\theta}$ space we appeal to the block structure of the covariance matrix so that we can make use of some 'tricks' to avoid ever storing this rather large C matrix. Essentially, utilising the construction of the large matrix C as the inverse of K^{-1} and R we can then exploit the block structure of K and make use of the Woodbury identity to rewrite C as a collection of diagonal and smaller matrices. Full details can be found in Appendix A.4.

The remaining hyperparameters of the model, namely σ_e^2 (the observation error), l (the correlation length) and σ^2 (the scale in the covariance matrix) are all assumed to be fixed in Ross and Dy (2013). We wish to also learn these parameters from the training data and so we apply the following methods. For the observational error σ_e^2 we use maximum likelihood estimation, where

$$\hat{\sigma}_e^2 = \frac{\sum_{i=1}^{N}\sum_{k=1}^{L} \mathrm{E}[\boldsymbol{Z}]_{ik}\left(x_i - \mathrm{E}\left[f_i^{(k)}\right]\right)^2}{\sum_{i=1}^{N}\sum_{k=1}^{L} \mathrm{E}[\boldsymbol{Z}]_{ik}}.$$

For the remaining GP hyperparameters, we adopt a pragmatic and computationally efficient approach. We anticipate the assumption of a similar covariance structure in each mode to be reasonable in practice, and therefore fit a single GP to a subset of the training data (to reduce computation time) using an MCMC scheme (similar to that mentioned in Section 4.2.5). This step is performed prior to the variational inference scheme. We then perform variational inference with all GP hyperparameters fixed at their posteriors means.

9.3.2 Application to SKMs

We wish to utilise the model of Ross and Dy (2013) in our calibration setting to gain some computational efficiency and so we simply use this in place of the DPMGP model from Chapter 7. Of course, a key requirement is to be able to use the surrogate output to predict at new points in $\boldsymbol{\theta}$ space. Predictive densities can be calculated by finding the univariate means and variances at each of the $\boldsymbol{\theta}$ points and conditioning on the training data where for each cluster (omitting cluster subscripts) we calculate

$$\mu_* = \boldsymbol{k}_*^\top (K + \sigma_e^2 I_N)^{-1} \bar{\boldsymbol{x}} \tag{9.5}$$

and

$$\sigma_*^2 = \sigma_e^2 + k_{**} - \boldsymbol{k}_*^\top (K + \sigma_e^2 I_N)^{-1} \boldsymbol{k}_*. \tag{9.6}$$

Here, K is the covariance matrix between all of the training data, $\boldsymbol{k}_*$ is the covariance between the training data and the point in $\boldsymbol{\theta}$ space where we wish to predict, k_{**} is the covariance between the point and itself (which is then the scale σ^2). Finally, $\bar{\boldsymbol{x}}$ is the weighted average at each point with i^{th} element as the weighted average of all observations at θ_i, with weights given by the probabilities r of being in the cluster. The predictive densities are then calculated as

$$f(x) = \sum_{k=1}^{L} \bar{r}_{\cdot k} N(x; \mu_{*k}, \sigma^2_{*k})$$

where $\bar{r}_{\cdot k}$ is the probability of being in cluster k. Thus, in order to emulate the output at a new point in $\boldsymbol{\theta}$ space, a cluster would first be simulated using the cluster probabilities $\bar{r}_{\cdot k}$ and then a normal variate would be simulated with the corresponding mean and variance, (9.5) and (9.6) respectively. For this variational approach we require setting the truncation parameter L since this uses the stick-breaking approach as discussed in Section 6.3.1. Due to the bimodal nature of the Schlögl system we expect the number of clusters to be fairly low and set this at $L = 10$. Following analysis in Ross and Dy (2013) we run the algorithm until a pre-specified number of iterations is reached where we initialise $\boldsymbol{Z}$ so that all observations are allocated to the same cluster. Another minor difference in the variational approach compared to the MCMC is that we assume a fixed concentration parameter $\alpha = 1$ which we expect may have only a minor effect on the inference.

Schlögl emulation

As in Chapter 8, we consider the task of inference for the rate constants in the Schlögl system, using data at discrete times. To allow comparison, we use the same data and priors as in Section 8.2. Likewise, the calibration scheme is as before (see Algorithm 5), with the DPMGP emulator replaced by the variational inference (VI) model.

Figure 9.1 shows marginal posterior densities for each (transformed) rate constant. The output of the calibration scheme is generally consistent with the true values that produced the data. The emulator gives state output at each time point that is consistent with the corresponding observation (see Figure 9.2). Despite this, some inconsistencies with the MJP posterior remain, as can be seen from the summaries in Table 9.1.

Nevertheless, posterior accuracy under the Ross and Dy (2013) surrogate (with VI) is comparable to that of the DPMGP surrogate (fitted exactly), yet computational efficiency is much improved. The total computational cost is reduced by a factor of 6 compared to using the MJP as the inferential model, and by a factor of almost 15 compared to using the DPMGP, as seen in Table 9.2.

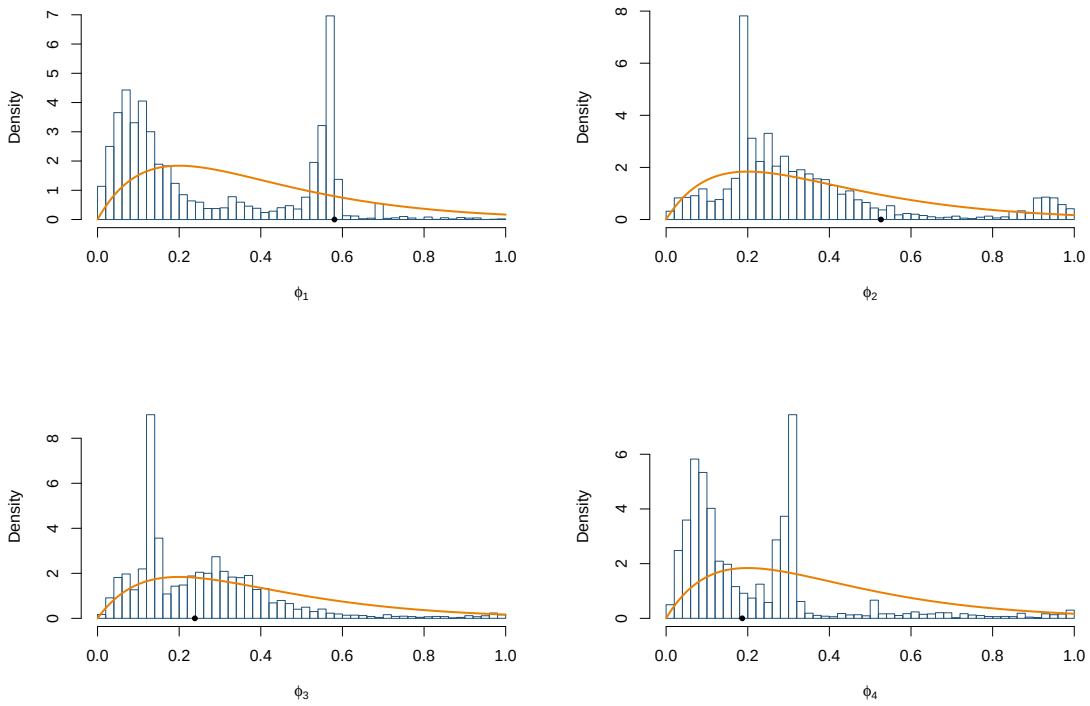

Figure 9.1: Histogram of posterior samples for each ϕ_i (at time $t = 20$) with prior (orange) overlaid and 'true' point (black).

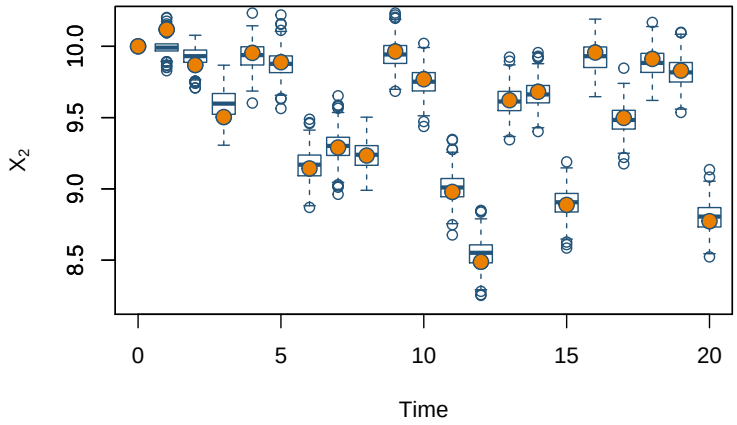

Figure 9.2: Boxplot of system data $\boldsymbol{x}$ particles at each time point with observation overlaid (orange).

Model	ϕ_1	ϕ_2	ϕ_3	ϕ_4
MJP	0.503 (0.116)	0.451 (0.108)	0.226 (0.058)	0.144 (0.085)
DPMGP ($M = 10$)	0.388 (0.194)	0.134 (0.167)	0.401 (0.184)	0.470 (0.238)
DPMGP ($M = 20$)	0.522 (0.134)	0.203 (0.162)	0.197 (0.180)	0.477 (0.218)
DPMGP ($M = 40$)	0.338 (0.112)	0.181 (0.132)	0.127 (0.074)	0.088 (0.079)
VI ($M = 40$)	0.280 (0.223)	0.333 (0.228)	0.252 (0.172)	0.214 (0.184)

Table 9.1: Table of marginal posterior means with standard deviations in parentheses.

Model	Run time (hours)	Training time (hours)
MJP	63	—
DPMGP ($M = 10$)	14	46
DPMGP ($M = 20$)	15	96
DPMGP ($M = 40$)	15	145
VI ($M = 40$)	11	<1

Table 9.2: Computation cost for each model.

Chapter 10

Conclusion

The intention of this thesis was to develop an emulator for stochastic kinetic models based on a mixture of Gaussian processes. Furthermore, we required this mixture to be flexible enough so that the number of components was not fixed. Using the emulator as part of an inference scheme then allows for the parameters of stochastic kinetic models to be inferred.

We began by introducing stochastic kinetic models (SKMs), chemical equations and also the Markov jump process representation of SKMs. Exact simulation methods were explored, specifically looking at Gillespie's direct method (Gillespie, 1976) before moving on to look at approximations such as the chemical Langevin equation and linear noise approximation. These approximations ignore the discreteness of the MJP but not the inherent stochasticity. The resulting computational efficiency of these schemes has been exploited by Golightly and Wilkinson (2005) and Komorowski et al. (2009) among many others, yet they are unable to successfully accomodate behaviours such as bifurcation. Three examples of SKMs were given, each one increasing in complexity (number of reactions and species). The birth-death model is a simple case with just one specie and two reactions. The Lotka-Volterra model (Lotka (1910), Volterra (1926)) describes the interaction between two species (predator and prey) through three reactions and exhibits interesting auto-regulatory behaviour for certain parameter choices. The final SKM we described was the Schlögl system (Schlögl, 1972); a system which, under certain conditions, exhibits bimodal stationarity.

In Chapter 3, after giving an overview of the Bayesian framework including Markov chain Monte Carlo (MCMC) methods, we considered likelihood-free methods which can be utilised for inferring the rate constants of SKMs. Likelihood-free MCMC was one such method but often suffers from small acceptance rates in the Metropolis-Hastings step. As a way to overcome this problem, Sequential Monte Carlo (SMC) methods were then described, in particular the Bootstrap filter (Gordon et al. (1993), Cappé et al. (2007)),

where our posterior beliefs are updated as each observation (in this case in time t) becomes available. All of these methods would require simulating from the SKM of interest by using Gillespie's direct method. However, this is likely to be computationally prohibitive in cases where many reactions occur between observation times. This necessitated the development of a cheap surrogate model.

Chapter 4 gave an overview of Gaussian processes (GPs) and how they can be used for regression. After defining GPs and looking at the mean and covariance functions, we described how GP regression can be used for prediction and also how we can fit a GP to data by inferring the hyperparameters of the mean and covariance functions. After a simulation study at the end of this chapter, we then moved on to describe how GPs could be used in an emulation setting (Chapter 5) with emphasis on SKM output. Of course, if the output of the SKM of interest is fairly symmetric and unimodal then a single GP emulator would be sufficient as the cheap approximation method. However, when considering the bimodality of the Schlögl system a mixture of GPs would be more appropriate.

It is possible to build an emulator from a finite mixture of GPs and even place a prior on the number of components, but by placing a Dirichlet process prior on the number of components we have not only bypassed the need to specify the number *a priori* but also avoided placing an upper limit on how many components there can be. The Dirichlet process (DP) was introduced in Chapter 6 both formally and informally by looking at different representations such as the Chinese restaurant process and Pólya urn scheme. A Dirichlet process mixture (DPM) model (Ferguson (1973), Antoniak (1974)) has an infinite number of components in the mixture where only a finite number of which have observations allocated to them. We illustrated DPM models using two examples: a univariate mixture of normals and a mixture of regressions models. In each case we fitted the DPM using MCMC (Neal, 2000).

We required an emulator that approximates the SKM output as a function of the rate constants and hence need to account for spatial dependence (in the space of the rate constants) which we achieved through GP regression. The emulator also needed to be flexible such that the number of components needed could increase or descrease as necessary. Thus, by combining the ideas of GPs and DPMs we built a Dirichlet process mixture of Gaussian processes (DPMGP) in Chapter 7. We described how we could evaluate certain parts of the likelihood more efficiently as well as how each data point is assigned to a cluster through the approach of Neal (2000). We then showed how the model fits to data simulated from a two component mixture of GPs and we were able to adequately account for the bimodal behaviour of the observed data.

In Chapter 8 the DPMGP model was used as an emulator for the Schlögl system. The

multi-modal output from this system proved to be a good test case for using the DPMGP as an emulator since the output is fairly complex, and can not be modelled with just a single GP. We inferred the posterior distribution of the rate constants by first running the Bootstrap filter with Gillespie's direct method. This allowed exact (simulation-based) inference of the rate constants, and provided a benchmark with which to compare competing, albeit approximate, methods. We then investigated the DPMGP approximation by using the fitted emulator as part of an SMC scheme to infer the rate constants. We explored different levels of approximation by changing the number of replicate observations at each point in $\boldsymbol{\theta}$-space. We found, as expected, the posteriors to be closer to the 'true' posteriors (obtained by Gillespie's direct method) as the number of replicate observations increased, thereby making the emulator more accurate. This of course came with more computational cost in terms of training the emulator. Specifically, we found that the total CPU time for training the emulator with 10 replicate observations at each point was 46 hours, increasing to 96 hours with 20 replicates and finally 145 hours with 40 replicates. The inference time however was not affected as much with all three emulators taking around 15 hours. The emulator was more efficient overall than the MJP model only in the case of 10 replicates, with a total run time of 60 hours (compared to 63 hours with the MJP).

Because of the computational cost of fitting the emulator to a large amount of points (which involves allocating each point individually in serial) we began exploring an approximate inference scheme known as variational inference (Chapter 9). We began the chapter by giving an overview of variational inference and then moved on to look at a slightly different DPMGP model introduced by Ross and Dy (2013). We then used the VI approximation as the emulator model for output from the Schlögl system, just as we did in Chapter 8 with the DPMGP. The marginal posterior densities obtained here were consistent with the true values used to simulate the observations. The loss in accuracy here was due to the further approximation being made by using variaional inference. However, the payoff here means that the computational cost is reduced much further when compared to using Gillespie's direct method in the inference scheme.

10.1 Future work

There are many possible extensions to the work in this thesis. Most notably, a more in-depth study of inference for the Schlögl system would be desired. Thus far, inference was based on a relatively small selection of training data. With improvements made in terms of efficiency due to the variational approach, the training set could be expanded in $\boldsymbol{\theta}$ space as well as the number of replicates. It would be expected that the posterior distribution

obtained with a larger training set would be even more similar to the posterior obtained using the MJP model. Nevertheless, further experiments are required to determine an adequate balance between computational efficiency and calibration accuracy.

Inference for the Schlögl system was based on data at 20 time points for the second species. A possible route to explore could be performing inference using data at more or fewer points in time in order to see the effect this has on the approximate posterior obtained. Another possibility could be performing inference given data of a different species of the system.

Although we have focused on the use of a DPMGP emulator for calibration of stochastic kinetic models, we note its potential uses for calibration of a wider class of models. Many complex stochastic processes exhibit dynamics that are not well described by a single GP. For example the double–well diffusion process (Buffett et al., 2013) is ubiquitously used to model relative paleointensities which provide estimates of the strength of the Earths' axial dipole and polarity (Morzfeld et al., 2017). Typical datasets used to calibrate such models contain thousands of observations, motivating the need for efficient inference schemes such as the one considered here.

It may also be possible to exploit the computational efficiency of the surrogate, whilst exactly targeting the posterior under the MJP, by using a delayed acceptance MCMC scheme (e.g. Golightly et al. (2015)). Essentially, in an initial screening stage, proposals are tried under the surrogate, and the expensive MJP simulator is only run for those proposals that pass this initial step. Since the likelihood under the MJP is intractable, this approach will require use of a pseudo-marginal Metropolis-Hastings scheme (see e.g. Andrieu et al. (2010)). This is a promising avenue for future research.

Appendix A

Miscellaneous

A.1 Probability density functions

A.1.1 Inverse gamma distribution

A positive random variable X has an inverse gamma distribution, denoted $X \sim IG(\alpha, \beta)$, with shape parameter α and scale parameter β with probability density function

$$\pi(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}(1/x)^{\alpha+1} \exp(-\beta/x).$$

A.1.2 Lognormal distribution

A positive random variable X has a lognormal distribution, denoted $X \sim LN(\mu, \sigma^2)$, with probability density function

$$\pi(x) = \frac{1}{x}\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\log x - \mu)^2}{2\sigma^2}\right).$$

A.2 Some useful multivariate normal (MVN) results

Characterisation: $\boldsymbol{Y} = (Y_1, \ldots, Y_q)^\top$ has a multivariate normal distribution with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$, denoted by $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $N_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if its density is

$$p(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{q/2}|\boldsymbol{\Sigma}|^{1/2} \exp\left\{-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})\right\}.$$

Linear transformations: if $\boldsymbol{A}$ is a $r \times q$ matrix of constants and $\boldsymbol{b}$ is a r-dimensional vector

of constants and $\boldsymbol{Y} \sim N_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then

$$\boldsymbol{AY} + \boldsymbol{b} \sim N_r(\boldsymbol{A\mu} + \boldsymbol{b}\,,\ \boldsymbol{A\Sigma A}^\top)$$

Marginal Distributions: if $\boldsymbol{Y}$ is divided into two blocks, $\boldsymbol{Y_1}$ containing the first q_1 components of $\boldsymbol{Y}$ and $\boldsymbol{Y_2}$ containing the other $q_2 = q - q_1$ components, then applying a partition of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the form

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \qquad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}$$

leads to $\boldsymbol{Y_i} \sim N_{q_i}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_{ii}), i = 1, 2$.

Conditional distributions: still using the same partitions on $\boldsymbol{Y}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$,

$$\boldsymbol{Y_1}|\boldsymbol{Y_2} = \boldsymbol{y_2} \sim N_{q_1}\left(\boldsymbol{\mu}_{1\cdot 2}, \boldsymbol{\Sigma}_{11\cdot 2}\right)$$

where $\boldsymbol{\mu}_{1\cdot 2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{y_2} - \boldsymbol{\mu}_2)$ and $\boldsymbol{\Sigma}_{11\cdot 2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}$. Similar results can be obtained for $\boldsymbol{Y_2}|\boldsymbol{Y_1}$ by exchanging the indices with 1 and 2.

Reconstruction of the joint distribution: if $\boldsymbol{Y_1}|\boldsymbol{Y_2} \sim N_{q_1}(\boldsymbol{\mu}_1 + \boldsymbol{B}_1[\boldsymbol{y_2} - \boldsymbol{\mu}_2], \boldsymbol{B}_2)$ for $q_1 \times q_2$ and $q_1 \times q_1$ matrices of constants $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ respectively and $\boldsymbol{Y_2} \sim N_{q_2}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ then

$$\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{Y_1} \\ \boldsymbol{Y_2} \end{pmatrix} \sim N_q\left[\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}\right]$$

where $\boldsymbol{\Sigma}_{11} = \boldsymbol{B}_2 + \boldsymbol{B}_1\boldsymbol{\Sigma}_{22}\boldsymbol{B}_1^\top$ and $\boldsymbol{\Sigma}_{21}^\top = \boldsymbol{\Sigma}_{12} = \boldsymbol{B}_1\boldsymbol{\Sigma}_{22}$.

A.3 Dirichlet process mixture of Gaussian processes

A.3.1 Log-likelihood function

Using efficient methods to calculate the inverse and determinant of the large $n \times n$ correlation matrix, we can efficiently evaluate the log-likelihood function. The data is stacked as $(x_1(\theta_1), x_2(\theta_1), \ldots, x_{n_1}(\theta_1), \ldots, x_1(\theta_N), \ldots, x_{n_N}(\theta_N))^\top = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top$ where $\boldsymbol{x}_i$ is a $n_i \times 1$ vector of all observations assigned to the cluster at $\boldsymbol{\theta}_i$. The log-likelihood contribution of the cluster is given by (up to an additive constant)

$$
\begin{aligned}
&-2\log f\{\boldsymbol{x}_1(\theta_1), \ldots, \boldsymbol{x}_N(\theta_N)\} \\
&\quad = \log|\sigma^2(\nu I_n + H)| + \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}^\top \{\sigma^2(\nu I_n + H)\}^{-1} \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\} \\
&\quad = 2n\log\sigma + \log|\nu I_n + H| + \frac{1}{\sigma^2}\{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}^\top (\nu I_n + H)^{-1} \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\} \\
&\quad = 2n\log\sigma + \log|\nu I_n + JH_N J^\top| \\
&\qquad + \frac{1}{\sigma^2}\{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}^\top \{\nu^{-1} I_n - \nu^{-2} J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top\} \\
&\qquad\qquad \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\} \\
&\quad = 2n\log\sigma + n\log\nu + \log|I_N + \nu^{-1} D H_N| \\
&\qquad + \frac{1}{\nu\sigma^2}\{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}^\top \{I_n - \nu^{-1} J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top\} \\
&\qquad\qquad \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}.
\end{aligned}
$$

The final term can be simplified by noting that $J^\top(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top = (n_1\bar{x}_1, \ldots, n_N\bar{x}_N)^\top = D\bar{\boldsymbol{x}}$, where $\bar{\boldsymbol{x}} = (\bar{x}_1, \ldots, \bar{x}_N)^\top$ is the $N \times 1$ vector of means at each $\boldsymbol{\theta}$-point, and $J^\top J = D$.

To ease notation let $\underline{\boldsymbol{x}} = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top$.

$$
\begin{aligned}
&\{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\}^\top \{I_n - \nu^{-1} J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top\} \{(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - JX\boldsymbol{\beta}\} \\
&\quad = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top - \nu^{-1}(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top) J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top \\
&\qquad\quad + (JX\boldsymbol{\beta})^\top JX\boldsymbol{\beta} - \nu^{-1}(JX\boldsymbol{\beta})^\top J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top JX\boldsymbol{\beta} \\
&\qquad\qquad - 2(\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top) JX\boldsymbol{\beta} + 2\nu^{-1}(JX\boldsymbol{\beta})^\top J(I_N + \nu^{-1} H_N D)^{-1} H_N J^\top (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_N^\top)^\top \\
&\quad = \underline{\boldsymbol{x}}^\top \underline{\boldsymbol{x}} - \nu^{-1} \bar{\boldsymbol{x}}^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D\bar{\boldsymbol{x}} + \boldsymbol{\beta}^\top X^\top DX\boldsymbol{\beta} \\
&\qquad - \nu^{-1}\boldsymbol{\beta}^\top X^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N DX\boldsymbol{\beta} - 2\bar{\boldsymbol{x}}^\top DX\boldsymbol{\beta} + \\
&\qquad\quad 2\nu^{-1}\boldsymbol{\beta}^\top X^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D\bar{\boldsymbol{x}} \\
&\quad = \underline{\boldsymbol{x}}^\top \underline{\boldsymbol{x}} + \boldsymbol{\beta}^\top X^\top DX\boldsymbol{\beta} - 2\bar{\boldsymbol{x}}^\top DX\boldsymbol{\beta} \\
&\qquad - \nu^{-1}\{\bar{\boldsymbol{x}}^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D\bar{\boldsymbol{x}} + \boldsymbol{\beta}^\top X^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N DX\boldsymbol{\beta} \\
&\qquad\quad - 2\boldsymbol{\beta}^\top X^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D\bar{\boldsymbol{x}}\} \\
&\quad = (\underline{\boldsymbol{x}} - JX\boldsymbol{\beta})^\top (\underline{\boldsymbol{x}} - JX\boldsymbol{\beta}) - \nu^{-1}(\bar{\boldsymbol{x}} - X\boldsymbol{\beta})^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D(\bar{\boldsymbol{x}} - X\boldsymbol{\beta}).
\end{aligned}
$$

Hence, the log-likelihood is given by (up to an additive constant)

$$
\begin{aligned}
&-2\log f\{\boldsymbol{x}_1(\theta_1), \ldots, \boldsymbol{x}_N(\theta_N)\} \\
&\qquad = 2n\log\sigma + n\log\nu + \log|I_N + \nu^{-1} DH_N| \\
&\qquad\qquad + \frac{1}{\nu\sigma^2}\Bigg\{(\underline{\boldsymbol{x}} - JX\boldsymbol{\beta})^\top (\underline{\boldsymbol{x}} - JX\boldsymbol{\beta}) \\
&\qquad\qquad\qquad - \nu^{-1}(\bar{\boldsymbol{x}} - X\boldsymbol{\beta})^\top D(I_N + \nu^{-1} H_N D)^{-1} H_N D(\bar{\boldsymbol{x}} - X\boldsymbol{\beta})\Bigg\}.
\end{aligned}
$$

A.3.2 Gibbs step for $\boldsymbol{\beta}$

The prior distribution for $\boldsymbol{\beta}|\sigma^2 \sim N(\boldsymbol{m}_0, \sigma^2 V_0)$ has density, for $\boldsymbol{\beta} \in \mathbb{R}^p$,

$$\pi(\boldsymbol{\beta}|\sigma^2) \propto \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{m}_0)^\top V_0^{-1}(\boldsymbol{\beta}-\boldsymbol{m}_0)\right\}.$$

Using Bayes' Theorem, the posterior density is

$$\begin{aligned}
\pi(\boldsymbol{\beta}|\sigma^2, \underline{\boldsymbol{x}}) &\propto \pi(\boldsymbol{\beta}|\sigma^2) f\{\boldsymbol{x}_1(\theta_1), \ldots, \boldsymbol{x}_N(\theta_N)\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{m}_0)^\top V_0^{-1}(\boldsymbol{\beta}-\boldsymbol{m}_0)\right\} \\
&\quad \times \exp\left\{-\frac{1}{2\nu\sigma^2}\Big[(\underline{\boldsymbol{x}}-JX\boldsymbol{\beta})^\top(\underline{\boldsymbol{x}}-JX\boldsymbol{\beta}) \right. \\
&\qquad \left. -\nu^{-1}(D\bar{\boldsymbol{x}}-DX\boldsymbol{\beta})^\top(I_N+\nu^{-1}H_ND)^{-1}H_N(D\bar{\boldsymbol{x}}-DX\boldsymbol{\beta})\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\Big[(\boldsymbol{\beta}-\boldsymbol{m}_0)^\top V_0^{-1}(\boldsymbol{\beta}-\boldsymbol{m}_0)+\nu^{-1}(\underline{\boldsymbol{x}}-JX\boldsymbol{\beta})^\top(\underline{\boldsymbol{x}}-JX\boldsymbol{\beta}) \right. \right. \\
&\qquad \left. \left. -\nu^{-2}(D\bar{\boldsymbol{x}}-DX\boldsymbol{\beta})^\top(I_N+\nu^{-1}H_ND)^{-1}H_N(D\bar{\boldsymbol{x}}-DX\boldsymbol{\beta})\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\Big[\boldsymbol{\beta}^\top V_0^{-1}\boldsymbol{\beta}+\boldsymbol{m}_0^\top V_0^{-1}\boldsymbol{m}_0-2\boldsymbol{\beta}^\top V_0^{-1}\boldsymbol{m}_0+\nu^{-1}\underline{\boldsymbol{x}}^\top\underline{\boldsymbol{x}}+\nu^{-1}\boldsymbol{\beta}^\top X^\top DX\boldsymbol{\beta} \right. \right. \\
&\qquad -2\nu^{-1}\bar{\boldsymbol{x}}^\top DX\boldsymbol{\beta}-\nu^{-2}\bar{\boldsymbol{x}}^\top D(I_N+\nu^{-1}H_ND)^{-1}H_ND\bar{\boldsymbol{x}} \\
&\qquad +2\nu^{-2}\boldsymbol{\beta}^\top X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_ND\bar{\boldsymbol{x}} \\
&\qquad \left. \left. -\nu^{-2}\boldsymbol{\beta}^\top X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_NDX\boldsymbol{\beta}\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\Big[\boldsymbol{\beta}^\top V_0^{-1}\boldsymbol{\beta}-2\boldsymbol{\beta}^\top V_0^{-1}\boldsymbol{m}_0+\nu^{-1}\boldsymbol{\beta}^\top X^\top DX\boldsymbol{\beta} \right. \right. \\
&\qquad -2\nu^{-1}\bar{\boldsymbol{x}}^\top DX\boldsymbol{\beta}+2\nu^{-2}\boldsymbol{\beta}^\top X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_ND\bar{\boldsymbol{x}} \\
&\qquad \left. \left. -\nu^{-2}\boldsymbol{\beta}^\top X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_NDX\boldsymbol{\beta}\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}\Big[\boldsymbol{\beta}^\top\left(V_0^{-1}+\nu^{-1}X^\top DX-\nu^{-2}X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_NDX\right)\boldsymbol{\beta} \right. \right. \\
&\qquad \left. \left. -2\boldsymbol{\beta}^\top\left(V_0^{-1}\boldsymbol{m}_0+\nu^{-1}X^\top D\bar{\boldsymbol{x}}-\nu^{-2}X^\top D(I_N+\nu^{-1}H_ND)^{-1}H_ND\bar{\boldsymbol{x}}\right)\Big]\right\} \\
&\propto \exp\left\{-\frac{1}{2\sigma^2}(\boldsymbol{\beta}-\boldsymbol{m}^*)^\top V^{*-1}(\boldsymbol{\beta}-\boldsymbol{m}^*)\right\}
\end{aligned}$$

after completing the square, where

$$\begin{aligned}\boldsymbol{m}^* &= V^*\{V_0^{-1}\boldsymbol{m}_0 + \nu^{-1}X^\top D\bar{\boldsymbol{x}} - \nu^{-2}X^\top D(I_N + \nu^{-1}H_N D)^{-1}H_N D\bar{\boldsymbol{x}}\}\\ V^* &= \{V_0^{-1} + \nu^{-1}X^\top DX - \nu^{-2}X^\top D(I_N + \nu^{-1}H_N D)^{-1}H_N DX\}^{-1}.\end{aligned}$$

Thus the posterior is $\boldsymbol{\beta}|\sigma^2, \underline{\boldsymbol{x}} \sim N(\boldsymbol{m}^*, \sigma^2 V^*)$.

A.3.3 Metropolis-Hastings step for $\sigma^2, \nu, \boldsymbol{r}$

We perform a joint update on these three parameters due to the correlation that exists between them. The problem stems from the high correlation between σ^2 and ν and so we use the following change of variables $\nu' = \sigma^2\nu$ and perform the random walk on ν' as opposed to ν. We shall denote the vector of these parameters as $\boldsymbol{\psi} = (\sigma^2, \nu', \boldsymbol{r})$ which will be $(2+d)$-dimensional. Using Metropolis-Hastings with a proposal distribution $q(\boldsymbol{\psi}^*|\boldsymbol{\psi}) = LN(\log\boldsymbol{\psi}, \Omega)$, where Ω is the tuning parameter, the acceptance probability is $\alpha(\boldsymbol{\psi}^*|\boldsymbol{\psi}) = \min(1, A)$, where

$$\begin{aligned}A &= \frac{\pi(\boldsymbol{\psi}^*)}{\pi(\boldsymbol{\psi})} \times \frac{f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}^*)}{f(\underline{\boldsymbol{x}}|\boldsymbol{\psi})} \times \frac{q(\boldsymbol{\psi}|\boldsymbol{\psi}^*)}{q(\boldsymbol{\psi}^*|\boldsymbol{\psi})}\\ &= \frac{\pi(\sigma^{*2})\pi_\nu(\nu'^*)\pi(\boldsymbol{r}^*)}{\pi(\sigma^2)\pi_\nu(\nu')\pi(\boldsymbol{r})} \times \frac{\sigma^2}{\sigma^{*2}} \times \frac{f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}^*)}{f(\underline{\boldsymbol{x}}|\boldsymbol{\psi})} \times \frac{\prod_{i=1}^{d+2}\psi_i^*}{\prod_{i=1}^{d+2}\psi_i}\end{aligned}$$

since $\pi_{\nu'}(\nu') = \pi_\nu(\nu')/\sigma^2$. Taking logs gives, up to an additive constant

$$\begin{aligned}\log A &= \log\pi(\sigma^{*2}) - \log\pi(\sigma^2) + \log\pi_\nu(\nu'^*) - \log\pi_\nu(\nu') + \log\pi(\boldsymbol{r}^*) - \log\pi(\boldsymbol{r})\\ &\quad + \log\sigma^2 - \log\sigma^{*2} + \log f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}^*) - \log f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}) + \sum_{i=1}^{d+2}(\log\psi_i^* - \log\psi_i)\\ &= \log\pi(\sigma^{*2}) - \log\pi(\sigma^2) + \log\pi_\nu(\nu'^*) - \log\pi_\nu(\nu') + \log\pi(\boldsymbol{r}^*) - \log\pi(\boldsymbol{r})\\ &\quad + \log f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}^*) - \log f(\underline{\boldsymbol{x}}|\boldsymbol{\psi}) + \sum_{i=2}^{d+2}(\log\psi_i^* - \log\psi_i).\end{aligned}$$

A.3.4 MCMC diagnostics

Trace plots were examined in order to determine that the chain was mixing well between the number of clusters.

$M = 10$ **replicates**

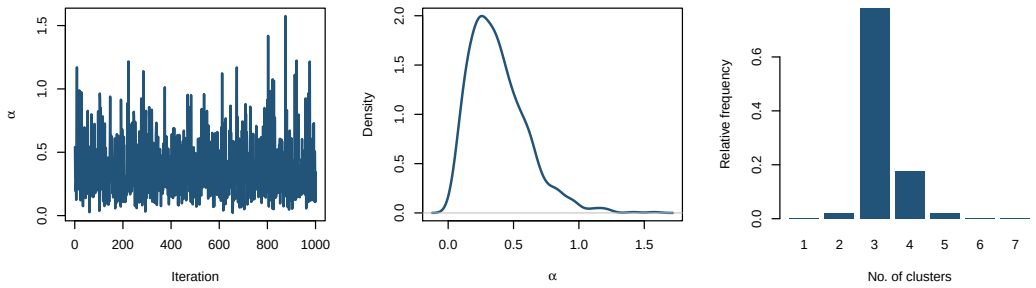

Figure A.1: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 1

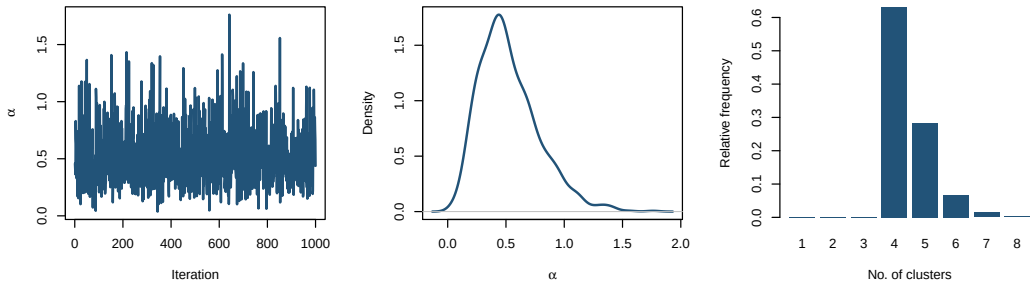

Figure A.2: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 2

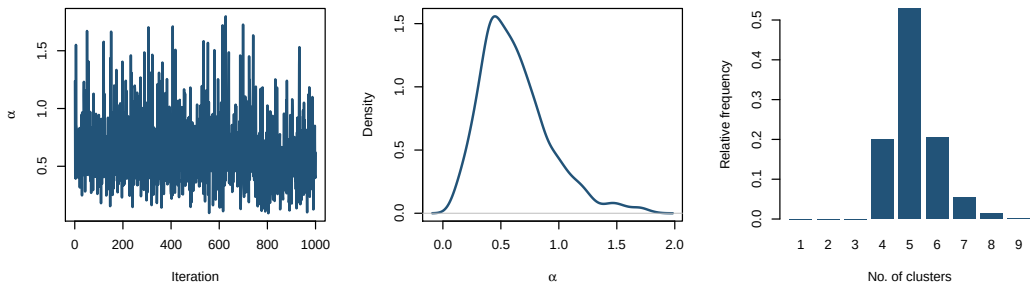

Figure A.3: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 3

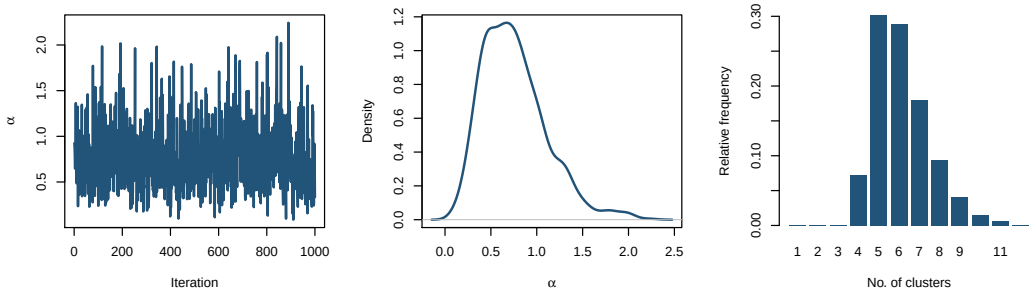

Figure A.4: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 4

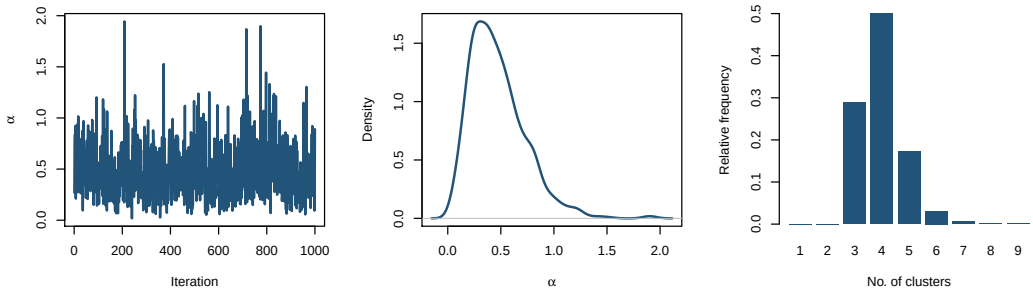

Figure A.5: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 5

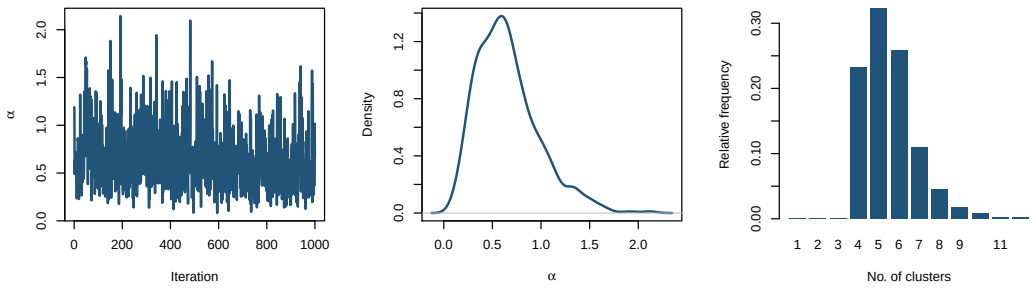

Figure A.6: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 6

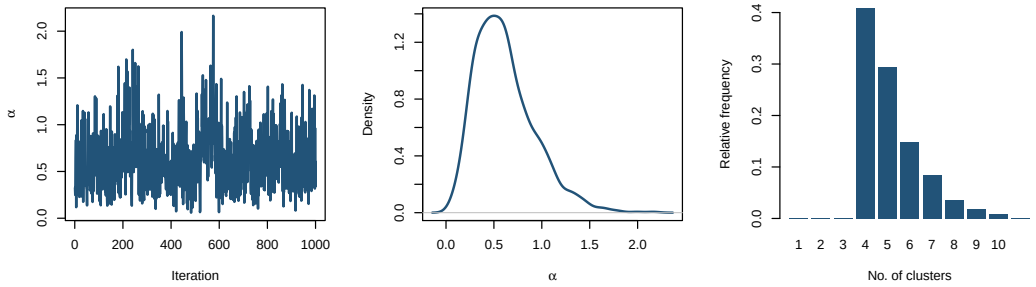

Figure A.7: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 7

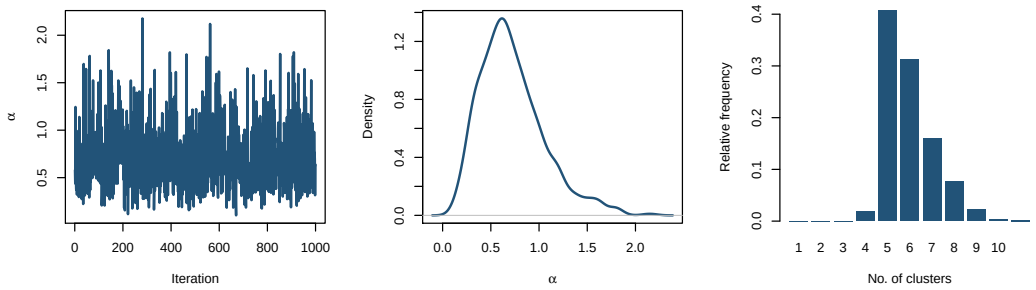

Figure A.8: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 8

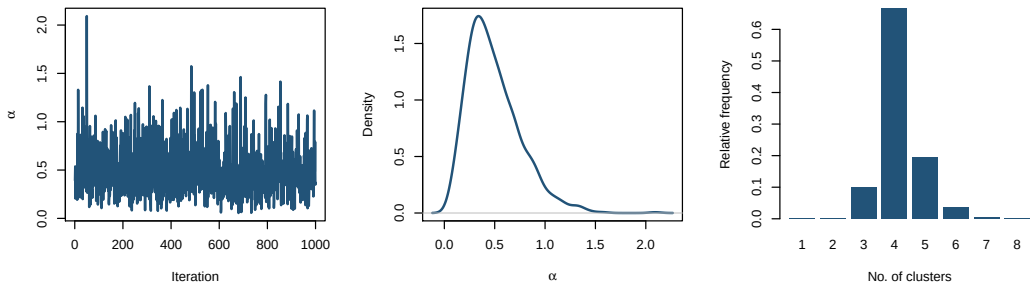

Figure A.9: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 9

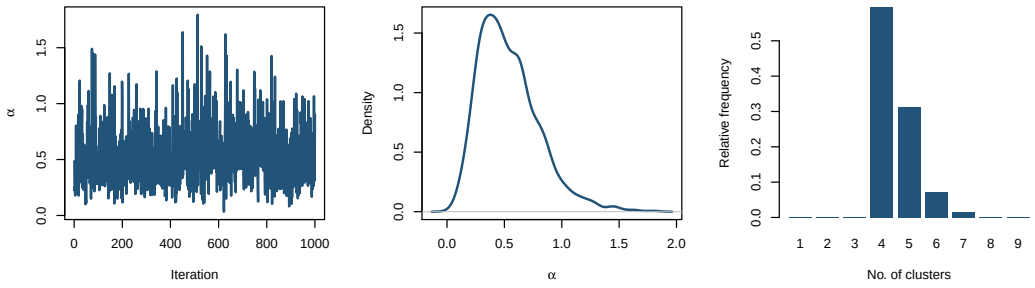

Figure A.10: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 10

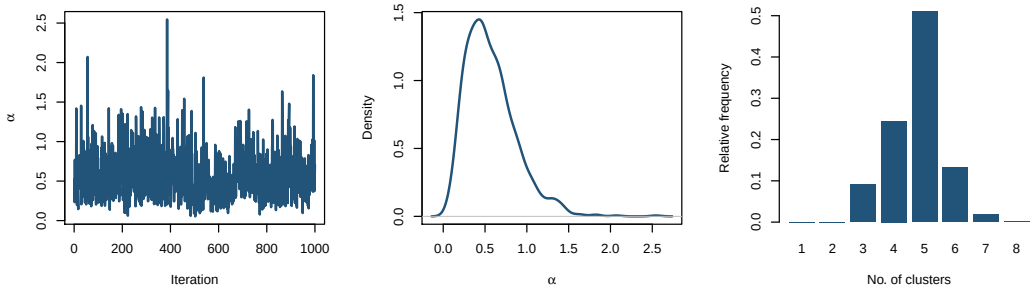

Figure A.11: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 11

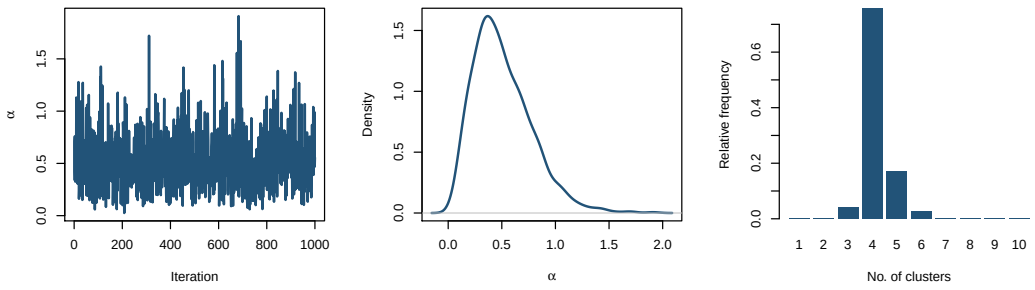

Figure A.12: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 12

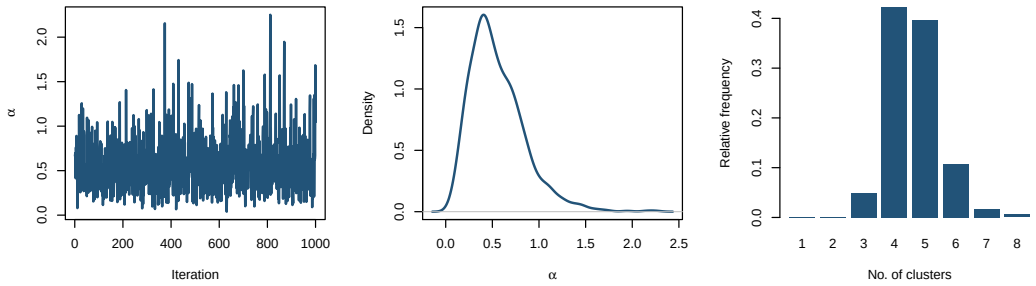

Figure A.13: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 13

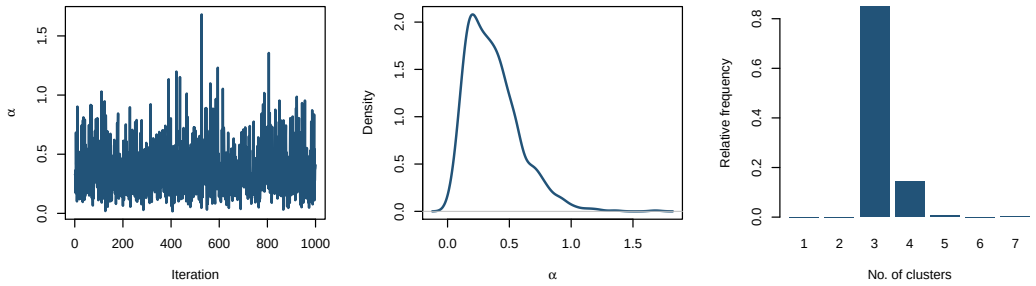

Figure A.14: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 14

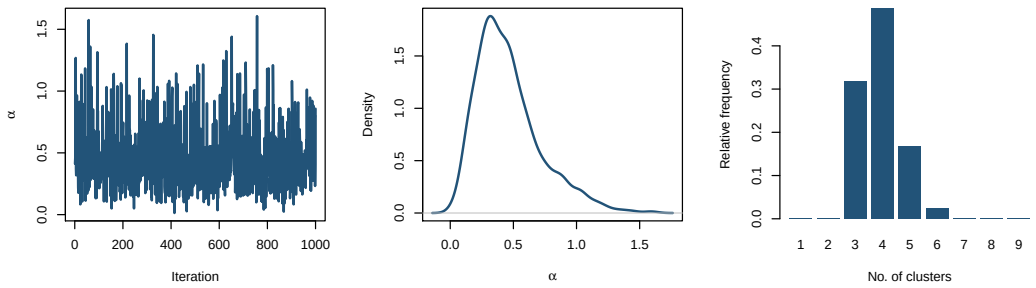

Figure A.15: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 15

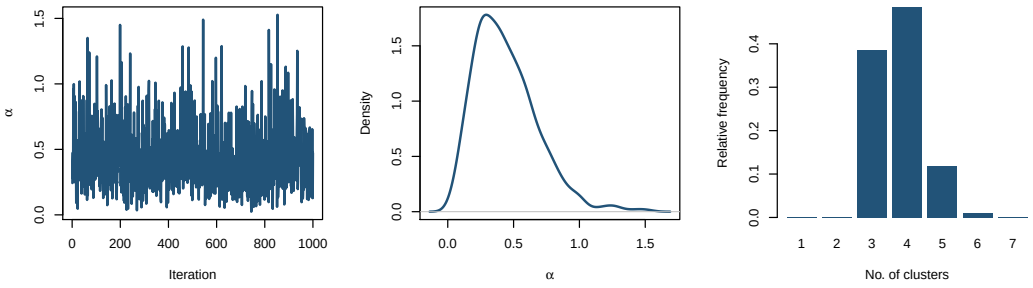

Figure A.16: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 16

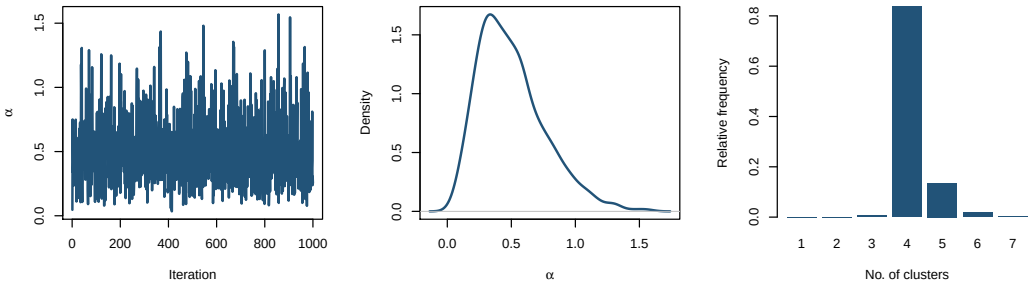

Figure A.17: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 17

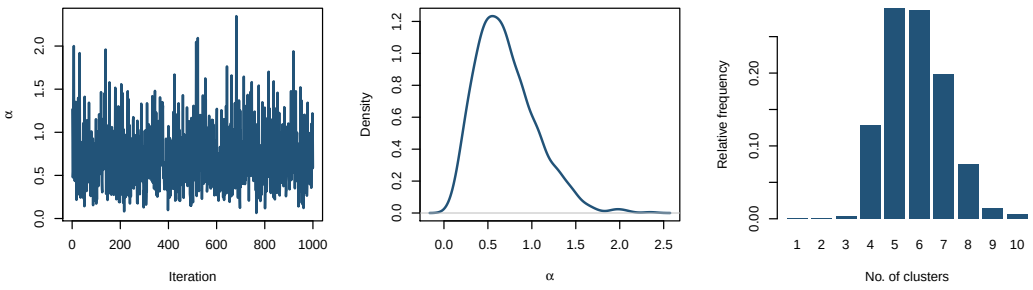

Figure A.18: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 18

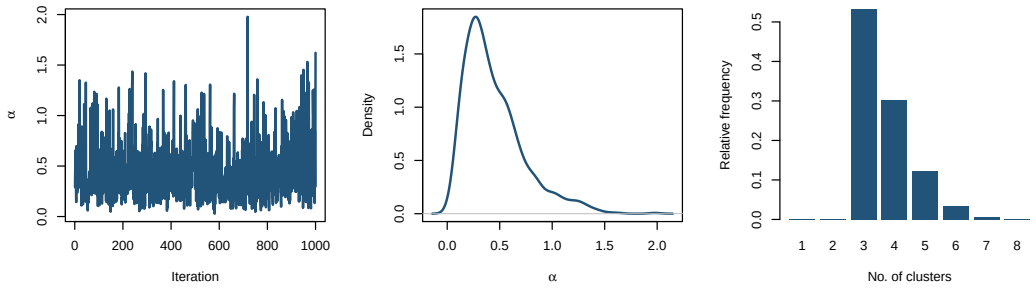

Figure A.19: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 19

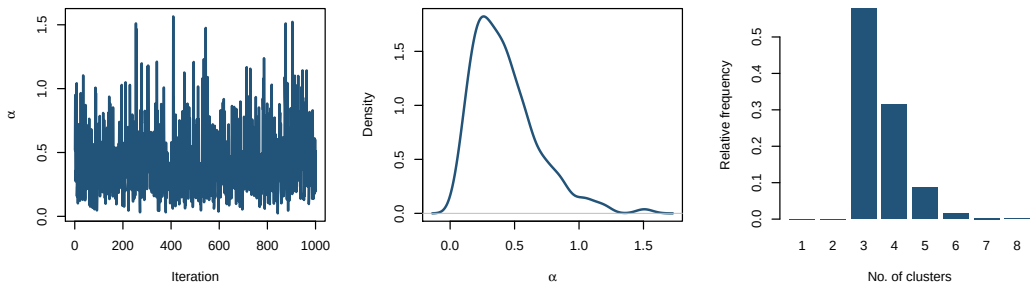

Figure A.20: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 20

$M = 20$ **replicates**

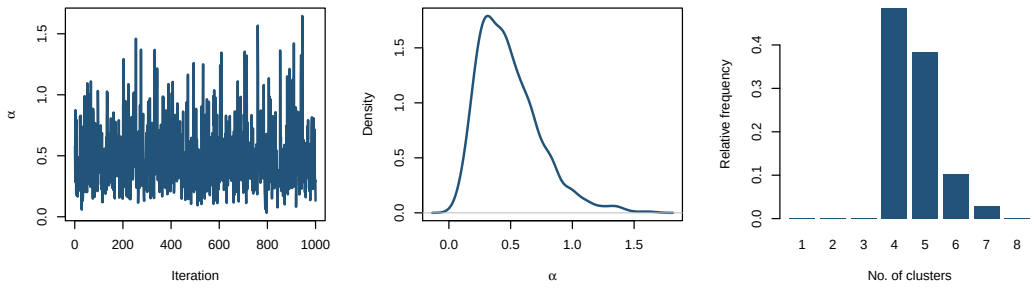

Figure A.21: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 1

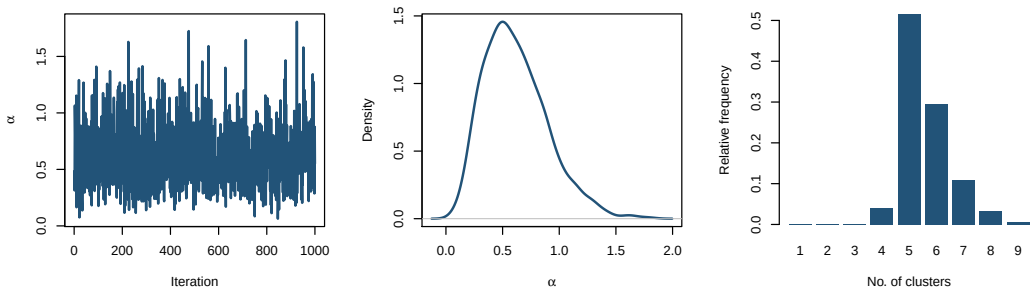

Figure A.22: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 2

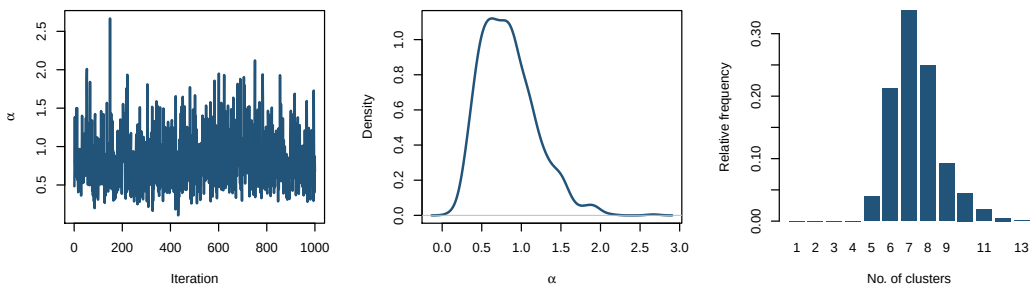

Figure A.23: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 3

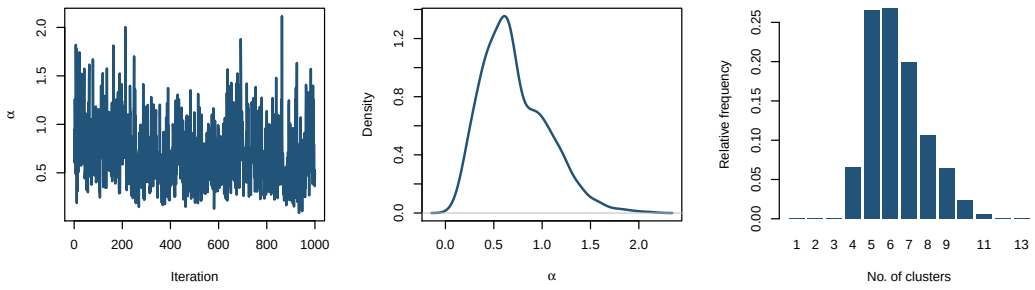

Figure A.24: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 4

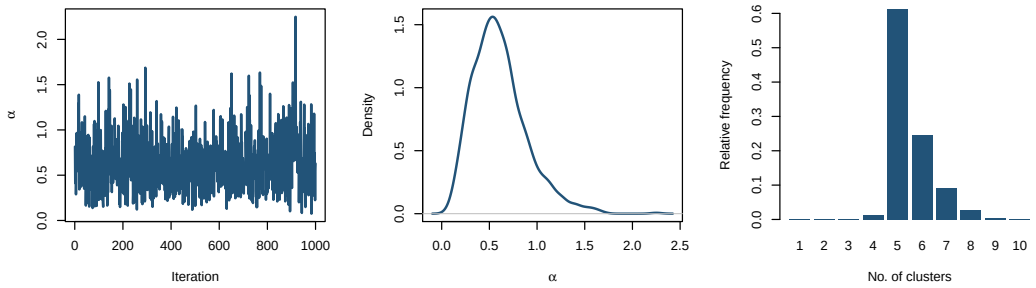

Figure A.25: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 5

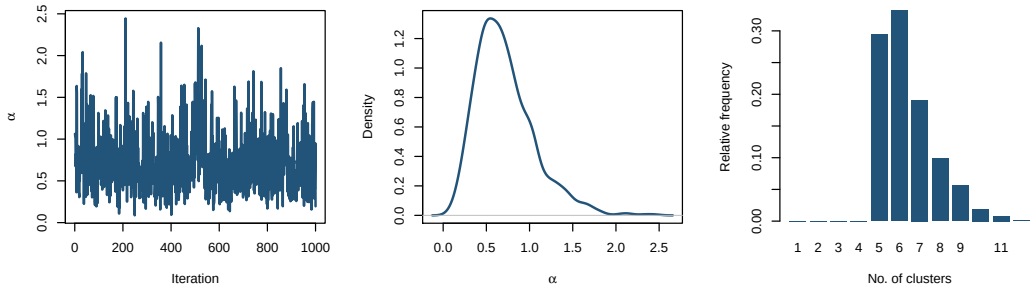

Figure A.26: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 6

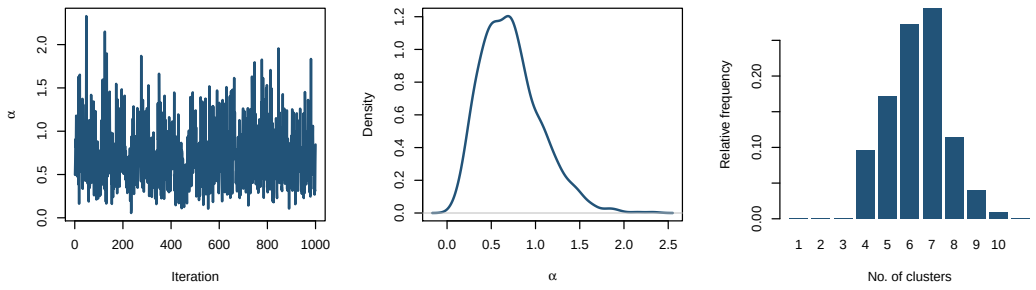

Figure A.27: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 7

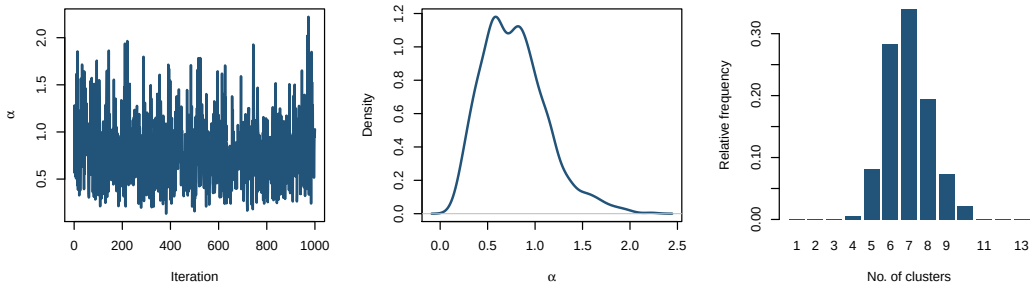

Figure A.28: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 8

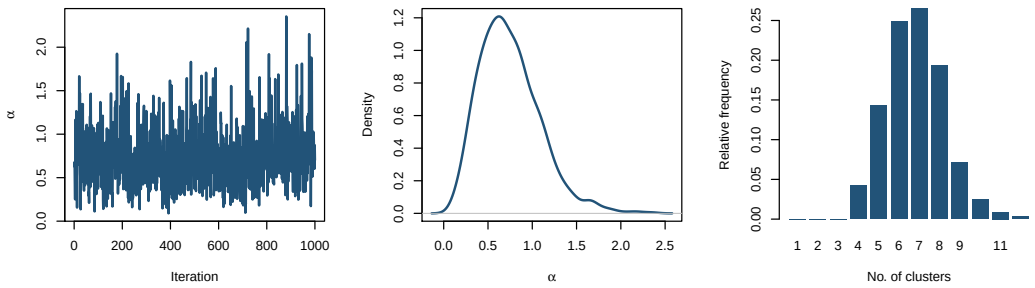

Figure A.29: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 9

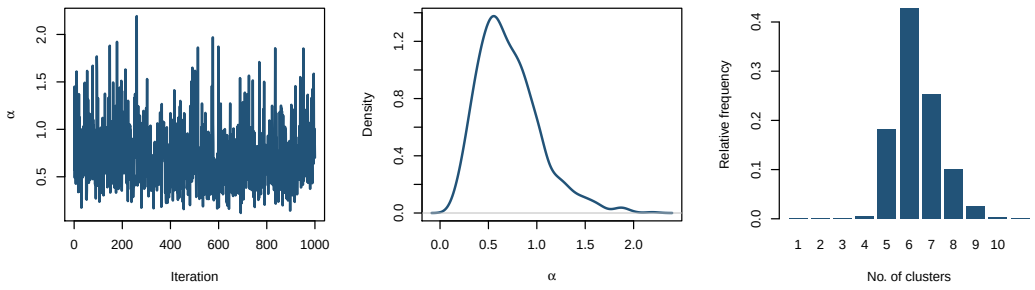

Figure A.30: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 10

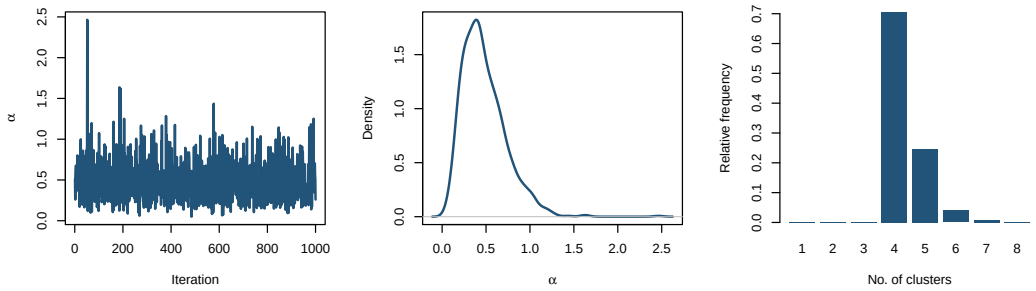

Figure A.31: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 11

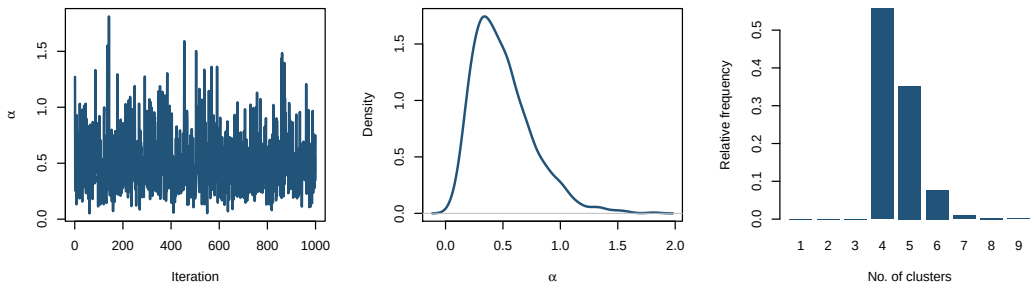

Figure A.32: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 12

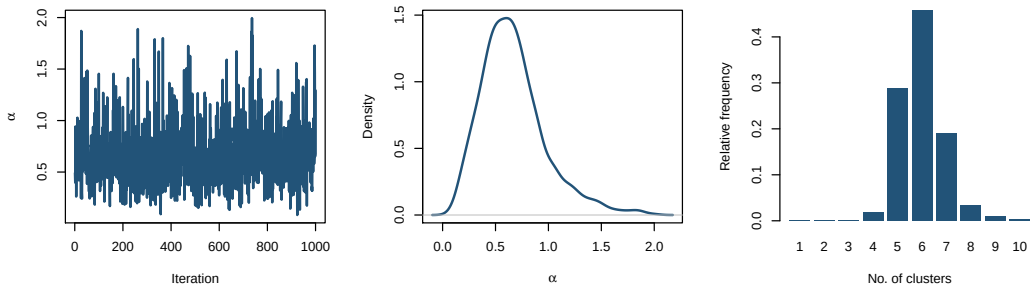

Figure A.33: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 13

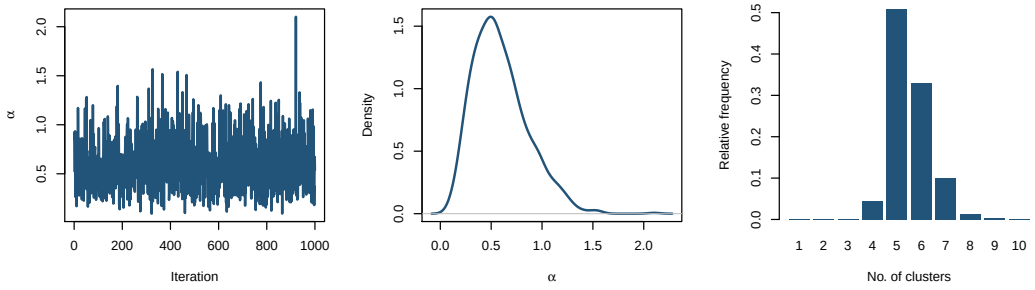

Figure A.34: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 14

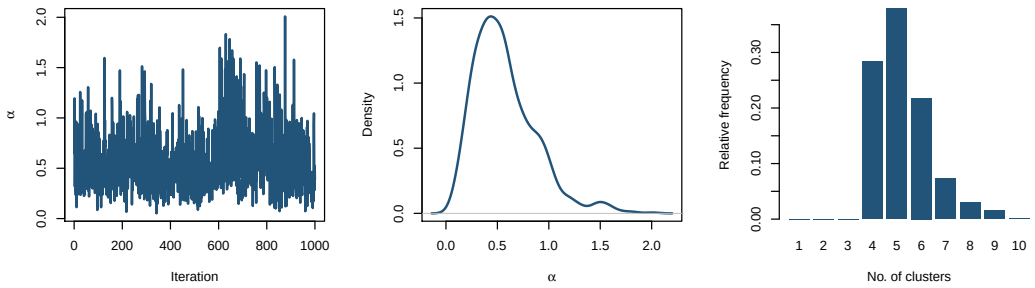

Figure A.35: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 15

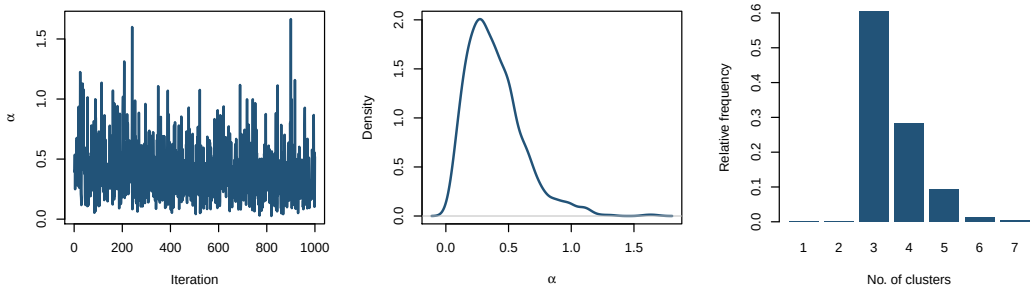

Figure A.36: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 16

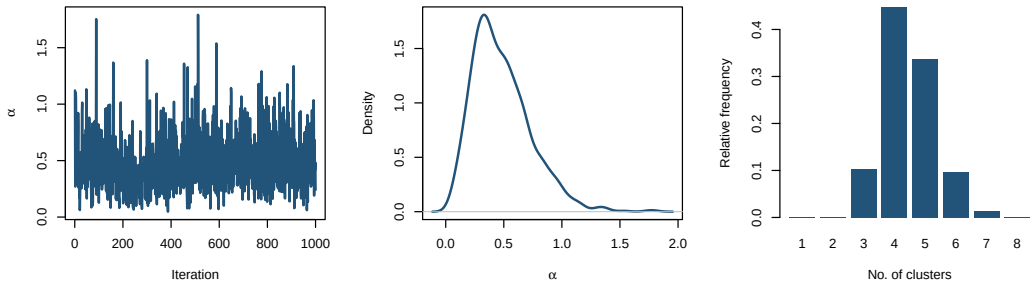

Figure A.37: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 17

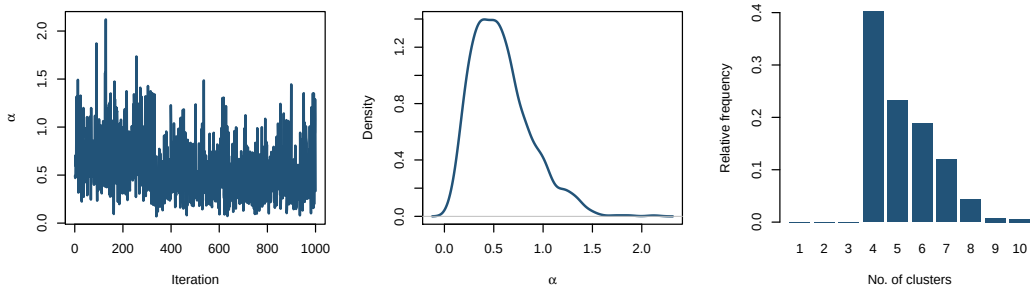

Figure A.38: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 18

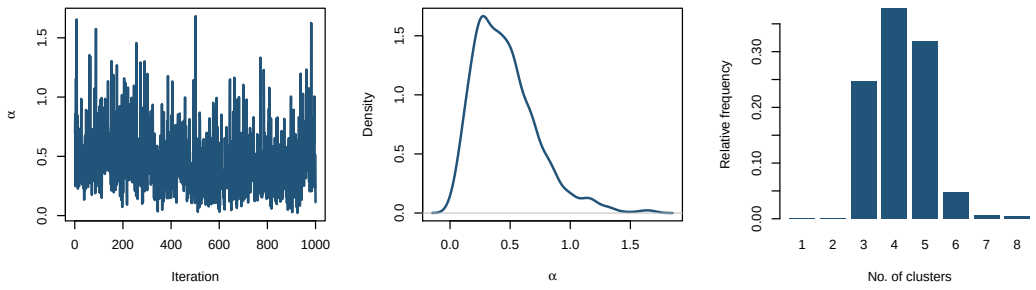

Figure A.39: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 19

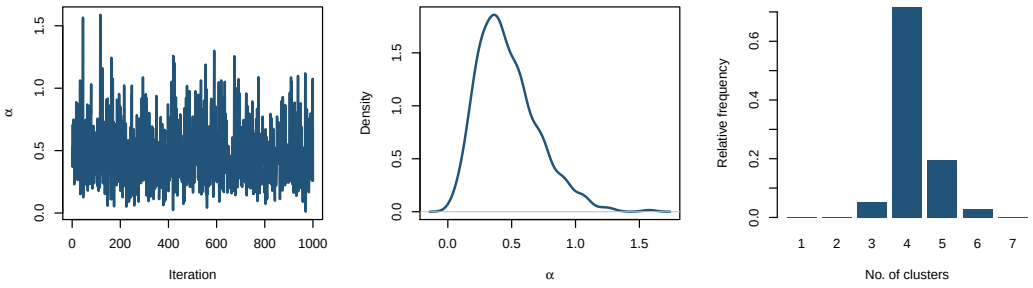

Figure A.40: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 20

$M = 40$ **replicates**

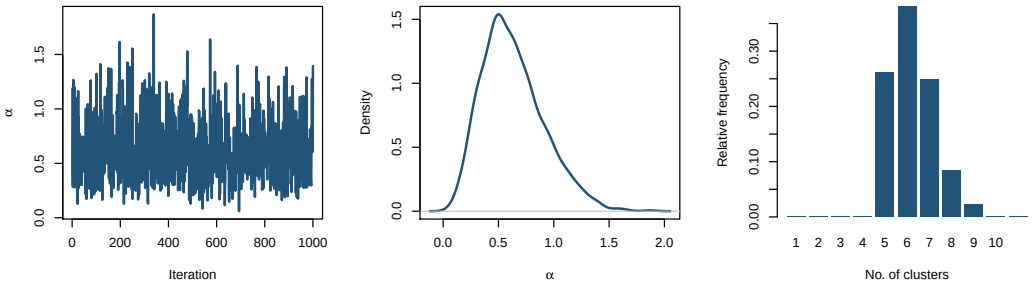

Figure A.41: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 1

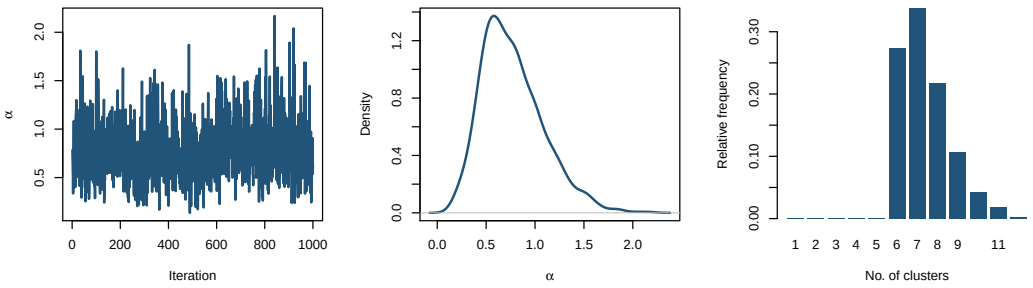

Figure A.42: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 2

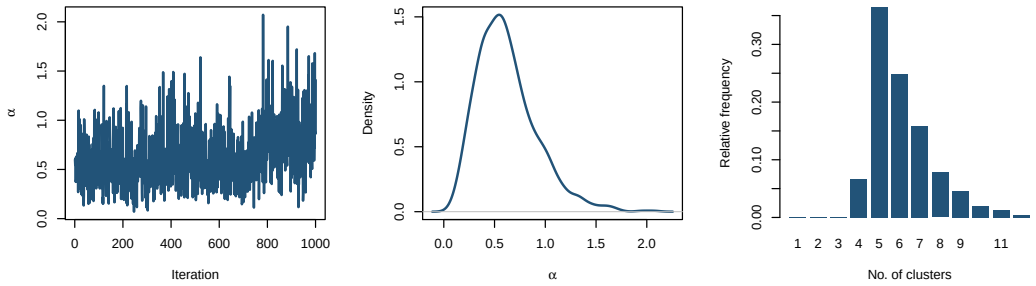

Figure A.43: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 3

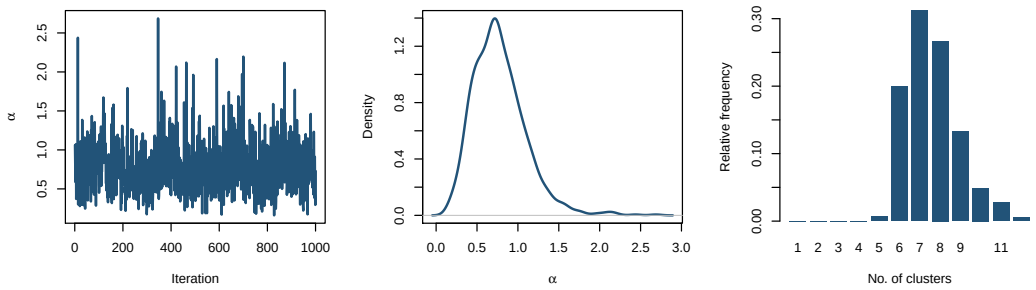

Figure A.44: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 4

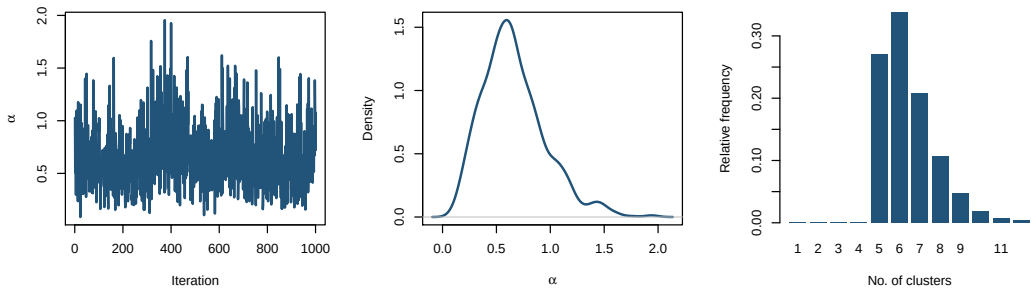

Figure A.45: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 5

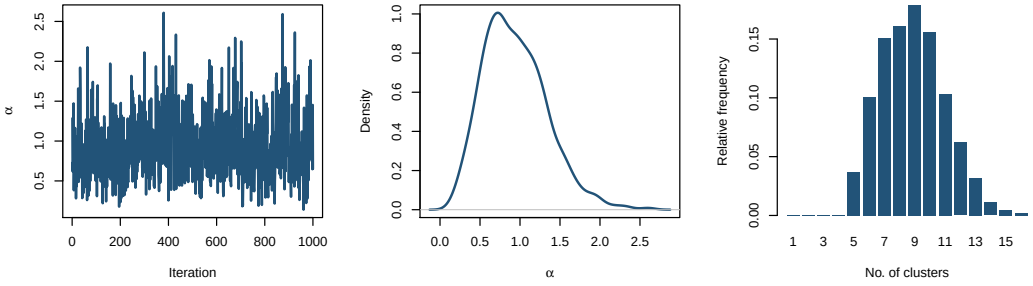

Figure A.46: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 6

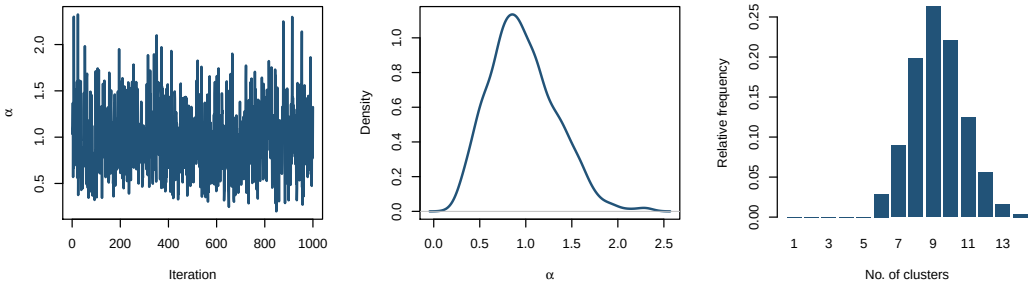

Figure A.47: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 7

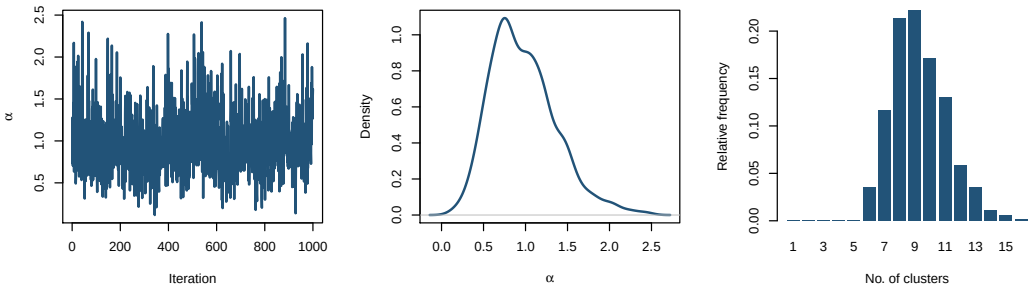

Figure A.48: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 8

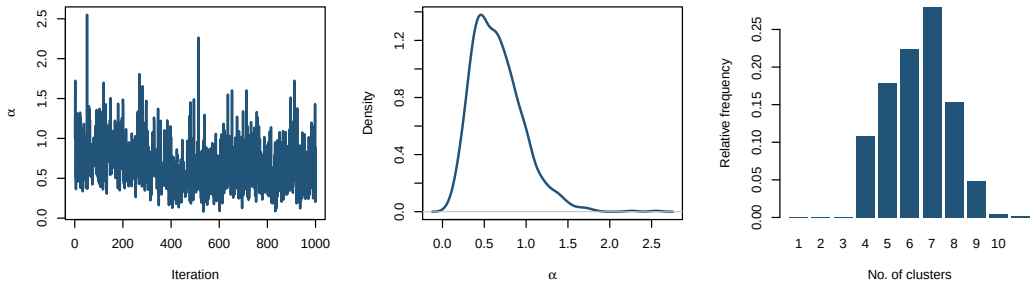

Figure A.49: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 9

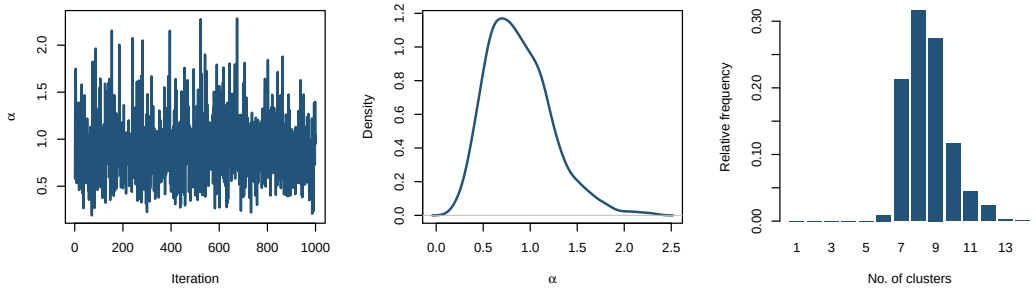

Figure A.50: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 10

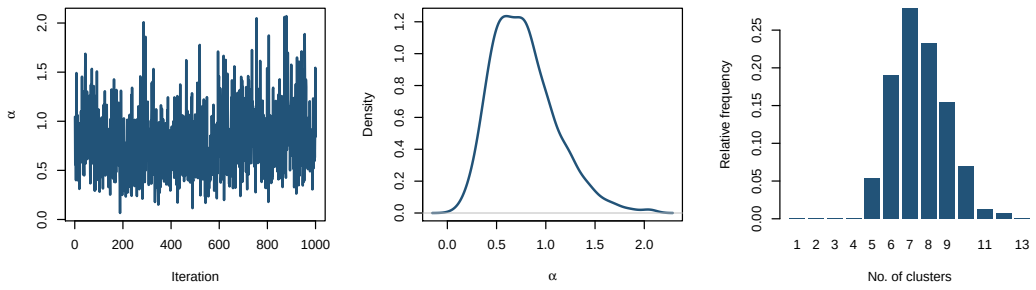

Figure A.51: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 11

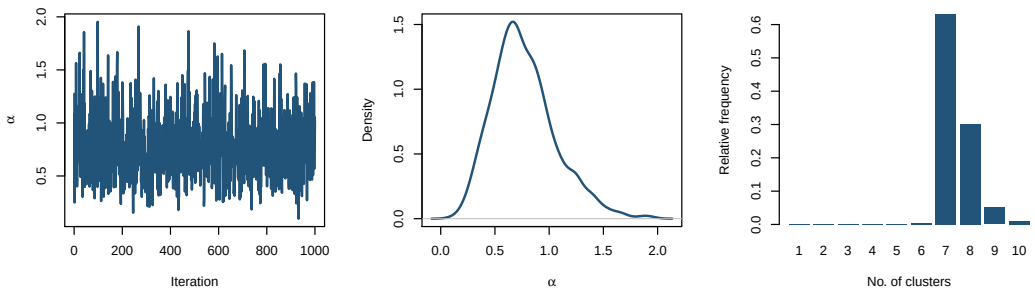

Figure A.52: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 12

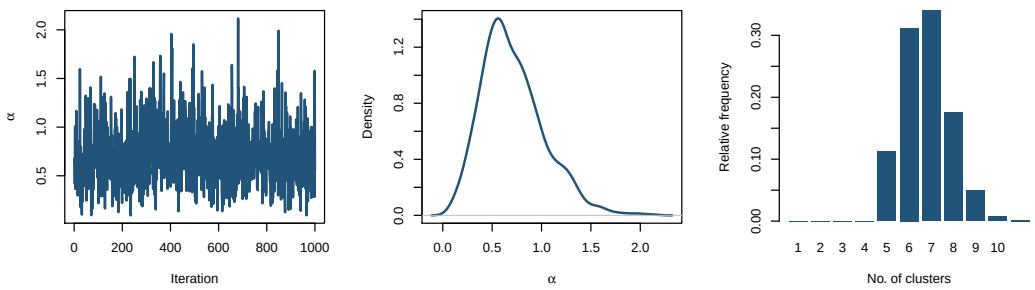

Figure A.53: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 13

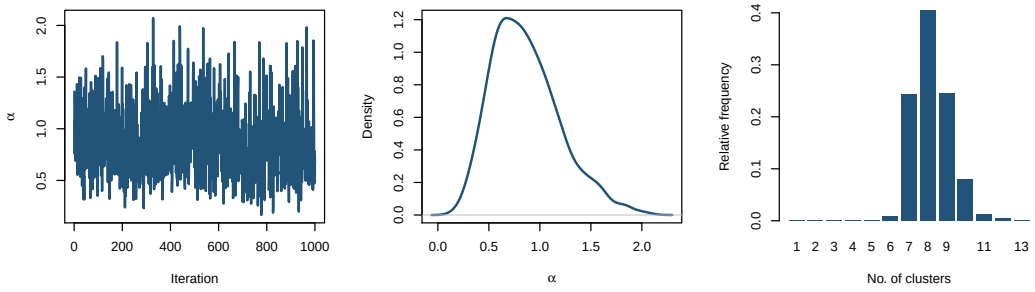

Figure A.54: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 14

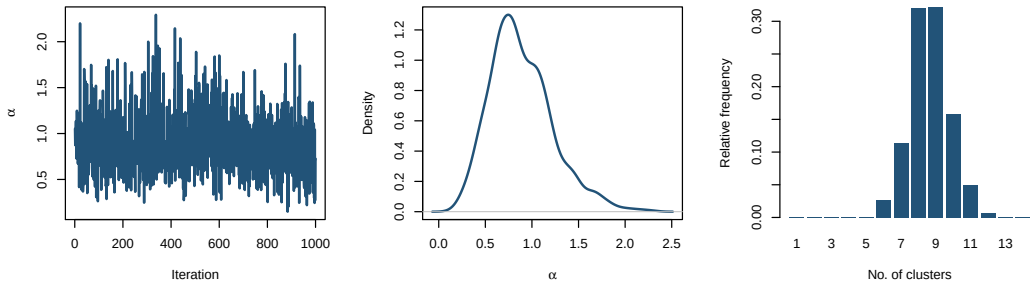

Figure A.55: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 15

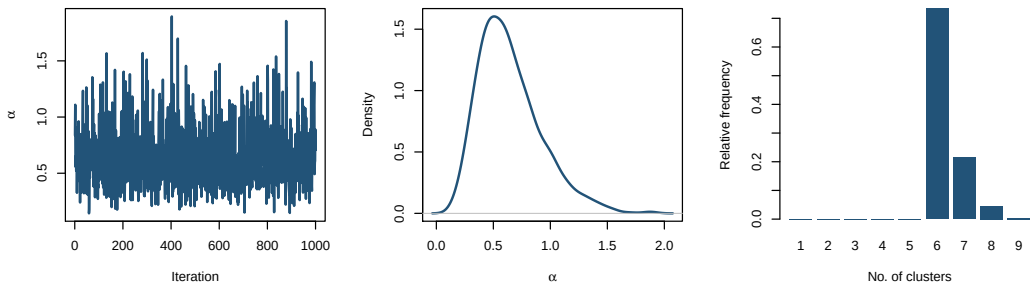

Figure A.56: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 16

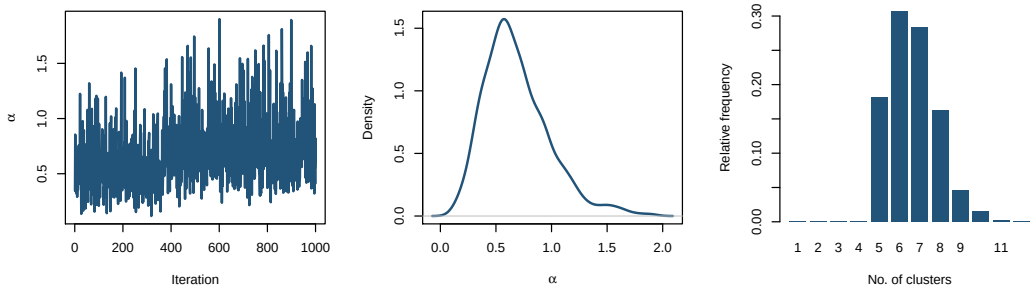

Figure A.57: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 17

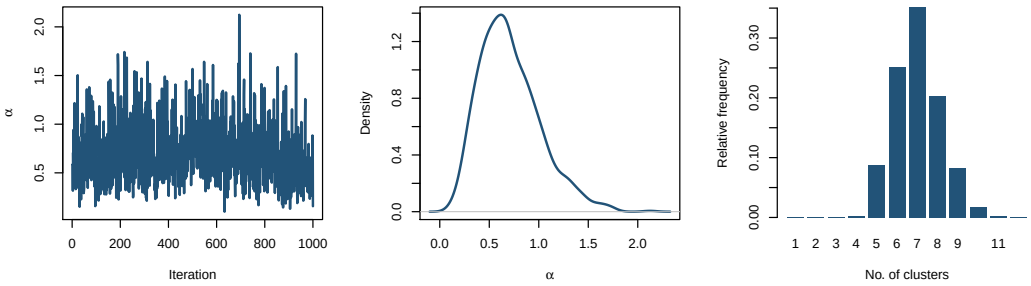

Figure A.58: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 18

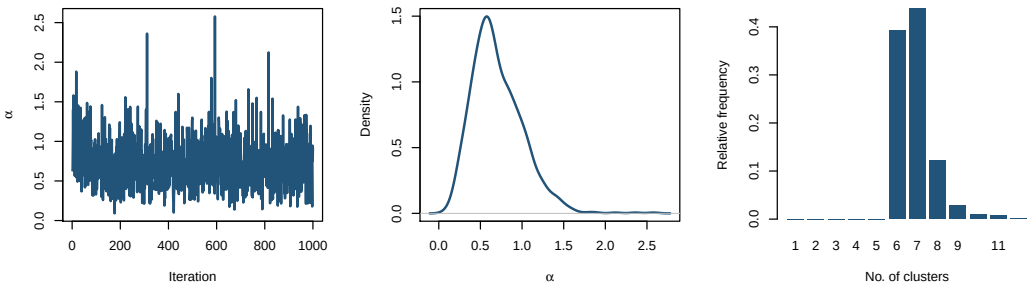

Figure A.59: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 19

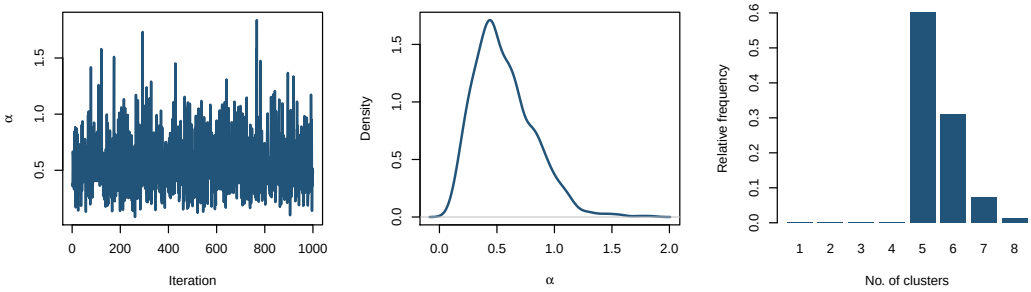

Figure A.60: Trace plot for α (left), marginal posterior density for α (middle) and relative frequency barplot for the number of cluster at time 20

A.4 Variational approach

The variational model as discussed in Section 9.3 requires working with quite large matrices. In particular, the matrix $C^{(k)} = (K^{(k)^{-1}} + R^{(k)})^{-1}$, for each k, is $NM \times NM$ in size which becomes very large when fitting to training data with a large amount of replicates (M) or locations (N). This section describes methods to avoid storing this matrix since we only need certain elements of it for calculations in the algorithm.

A.4.1 Calculating the mean

One part of the algorithm where we require the large matrix $C^{(k)}$ is during the calculation of the mean of each of the Gaussian processes. The mean for cluster k is

$$\boldsymbol{\mu}^{(k)} = C^{(k)} R^{(k)} \boldsymbol{x}$$

where

$$R^{(k)} = \frac{1}{\sigma_e^2} \text{diag}(\text{E}[\boldsymbol{Z}]_{1k}, \ldots, \text{E}[\boldsymbol{Z}]_{Nk}).$$

In the following derivation we drop the dependence on cluster number for ease of exposition, but in practice, the following quantities would have a subscript k to denote that they are calculated for cluster k. We start by rewriting $C = (K^{-1} + R)^{-1}$ by making use of the block structure of the covariance matrix $K = \nu I_{NM} + JK_N J^\top$, where J is as defined in Section 7.3 and K_N is the $N \times N$ covariance matrix. This gives

$$\begin{aligned} C &= (K^{-1} + R)^{-1} \\ &= \{(\nu I_{NM} + JK_N J^\top)^{-1} + R\}^{-1}. \end{aligned}$$

Now using the identity $(\nu I_{NM} + JK_N J^\top)^{-1} = \nu^{-1} I_{NM} - \nu^{-2} J(I_N + \nu^{-1} K_N \Delta)^{-1} K_N J^\top$ (shown in Section 7.3.1) gives

$$C = \{\nu^{-1} I_{NM} - \nu^{-2} J(I_N + \nu^{-1} K_N \Delta)^{-1} K_N J^\top + R\}^{-1}.$$

We can now apply the Woodbury identity (7.4) with $A = R + \nu^{-1} I_{NM}$ (an $NM \times NM$ diagonal matrix), $B = (I_N + \nu^{-1} K_N \Delta)^{-1}$ (an $N \times N$ matrix), $C = -\nu^{-2} J$ (an $NM \times N$ matrix) and $D = K_N J^\top$ (an $N \times NM$ matrix) where $\Delta = J^\top J = \text{diag}(M, M, \ldots, M)$ to

give

$$\begin{aligned}
C &= (R+\nu^{-1}I_{NM})^{-1} \\
&\quad + \nu^{-2}(R+\nu^{-1}I_{NM})^{-1}J[\{(I_N+\nu^{-1}K_N\Delta)^{-1}\}^{-1} \\
&\qquad\qquad - \nu^{-2}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1}J]^{-1}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1} \\
&= (R+\nu^{-1}I_{NM})^{-1} \\
&\quad + \nu^{-2}(R+\nu^{-1}I_{NM})^{-1}J\{(I_N+\nu^{-1}K_N\Delta) \\
&\qquad\qquad - \nu^{-2}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1}J\}^{-1}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1}.
\end{aligned}$$

Note that C is now formed using a collection of diagonal matrices and smaller $N \times N$ matrices, and so we can determine $\boldsymbol{\mu} = CR\boldsymbol{x}$ by right multiplying by $R\boldsymbol{x}$ to give

$$\begin{aligned}
\boldsymbol{\mu} &= (R+\nu^{-1}I_{NM})^{-1}R\boldsymbol{x} \\
&\quad + \nu^{-2}(R+\nu^{-1}I_{NM})^{-1}J\{(I_N+\nu^{-1}K_N\Delta) \\
&\qquad - \nu^{-2}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1}J\}^{-1}K_NJ^\top(R+\nu^{-1}I_{NM})^{-1}R\boldsymbol{x}.
\end{aligned}$$

This formulation gives a much more manageable approach to determining $\boldsymbol{\mu}$ without the need to store the large $NM \times NM$ matrix C: it only requires the calculation of large but diagonal matrices, $N \times N$ matrices and the sparse $NM \times N$ matrix J.

Bibliography

Aldous, D. J. (1985), Exchangeability and related topics, *in* 'École d'Été de Probabilités de Saint-Flour XIII1983', Springer, pp. 1–198.

Ale, A., Kirk, P. and Stumpf, M. P. (2013), 'A general moment expansion method for stochastic kinetic models', *The Journal of Chemical Physics* **138**(17), 174101.

Andrieu, C., Doucet, A. and Holenstein, R. (2010), 'Particle Markov chain Monte Carlo methods', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **72**(3), 269–342.

Antoniak, C. E. (1974), 'Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems', *The Annals of Statistics* **2**, 1152–1174.

Baggaley, A. W., Boys, R. J., Golightly, A., Sarson, G. R. and Shukurov, A. (2012), 'Inference for population dynamics in the Neolithic period', *The Annals of Applied Statistics* **6**(4), 1352–1376.

Bastos, L. S. and OHagan, A. (2009), 'Diagnostics for Gaussian process emulators', *Technometrics* **51**(4), 425–438.

Bayarri, M. J., Berger, J. O., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C.-H. and Tu, J. (2007), 'A framework for validation of computer models', *Technometrics* **49**(2), 138–154.

Beaumont, M. A., Zhang, W. and Balding, D. J. (2002), 'Approximate Bayesian computation in population genetics', *Genetics* **162**(4), 2025–2035.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer.

Blackwell, D. and MacQueen, J. B. (1973), 'Ferguson distributions via Pólya urn schemes', *The Annals of Statistics* **1**, 353–355.

Blei, D. M., Jordan, M. I. et al. (2006), 'Variational inference for Dirichlet process mixtures', *Bayesian Analysis* **1**(1), 121–143.

Blei, D. M., Kucukelbir, A. and McAuliffe, J. D. (2017), 'Variational inference: A review for statisticians', *Journal of the American Statistical Association* **112**(518), 859–877.

Boys, R. J., Wilkinson, D. J. and Kirkwood, T. B. (2008), 'Bayesian inference for a discretely observed stochastic kinetic model', *Statistics and Computing* **18**(2), 125–135.

Brooks, S., Gelman, A., Jones, G. and Meng, X.-L. (2011), *Handbook of Markov Chain Conte Carlo*, CRC press.

Buffett, B. A., Ziegler, L. and Constable, C. G. (2013), 'A stochastic model for palaeomagnetic field variations', *Geophysical Journal International* **195**(1), 86–97.
URL: *https://doi.org/10.1093/gji/ggt218*

Cappé, O., Godsill, S. J. and Moulines, E. (2007), 'An overview of existing methods and recent advances in sequential Monte Carlo', *Proceedings of the IEEE* **95**(5), 899–924.

Carnell, R. (2016), *lhs: Latin Hypercube Samples*. R package version 0.14.
URL: *https://CRAN.R-project.org/package=lhs*

Cooke, E. J., Savage, R. S., Kirk, P. D., Darkins, R. and Wild, D. L. (2011), 'Bayesian hierarchical clustering for microarray time series data with replicates and outlier measurements', *BMC Bioinformatics* **12**(1), 399.

Currin, C., Mitchell, T., Morris, M. and Ylvisaker, D. (1991), 'Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments', *Journal of the American Statistical Association* **86**(416), 953–963.

Elf, J. and Ehrenberg, M. (2003), 'Fast evolution of fluctuations in biochemical networks with the linear noise approximation', *Genome Res.* **13**(11), 2475–2484.

Escobar, M. D. and West, M. (1995), 'Bayesian density estimation and inference using mixtures', *Journal of the American Statistical Association* **90**(430), 577–588.

Fearnhead, P., Giagos, V. and Sherlock, C. (2014), 'Inference for reaction networks using the linear noise approximation', *Biometrics* **70**, 457–466.

Ferguson, T. S. (1973), 'A Bayesian analysis of some nonparametric problems', *The Annals of Statistics* pp. 209–230.

Gamerman, D. and Lopes, H. F. (2006), *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, 2nd edn, Chapman and Hall/CRC.

Gelman, A., Rubin, D. B. et al. (1992), 'Inference from iterative simulation using multiple sequences', *Statistical Science* **7**(4), 457–472.

Geman, S. and Geman, D. (1984), 'Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* (6), 721–741.

Gillespie, D. T. (1976), 'A general method for numerically simulating the stochastic time evolution of coupled chemical reactions', *Journal of Computational Physics* **22**(4), 403–434.

Gillespie, D. T. (1992), 'A rigorous derivation of the chemical master equation', *Physica A: Statistical Mechanics and its Applications* **188**(1-3), 404–425.

Gillespie, D. T. (2000), 'The chemical Langevin equation', *The Journal of Chemical Physics* **113**(1), 297–306.

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007), 'Probabilistic forecasts, calibration and sharpness', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **69**(2), 243–268.

Golightly, A. and Gillespie, C. S. (2013), Simulation of stochastic kinetic models, *in* 'In Silico Systems Biology', Springer, pp. 169–187.

Golightly, A., Henderson, D. A. and Sherlock, C. (2015), 'Delayed acceptance particle MCMC for exact inference in stochastic kinetic models', *Statistics and Computing* **25**(5), 1039–1055.
URL: *https://doi.org/10.1007/s11222-014-9469-x*

Golightly, A. and Sherlock, C. (n.d.), 'Efficient sampling of conditioned Markov jump processes', *Statistics and Computing* pp. 1–15.

Golightly, A. and Wilkinson, D. J. (2005), 'Bayesian inference for stochastic kinetic models using a diffusion approximation', *Biometrics* **61**(3), 781–788.

Golightly, A. and Wilkinson, D. J. (2006), 'Bayesian sequential inference for stochastic kinetic biochemical network models', *Journal of Computational Biology* **13**(3), 838–851.

Golightly, A. and Wilkinson, D. J. (2011), 'Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo', *Interface focus* pp. 807–820.

Gordon, N. J., Salmond, D. J. and Smith, A. F. (1993), Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *in* 'IEE Proceedings F-radar and signal processing', Vol. 140, IET, pp. 107–113.

Hager, W. W. (1989), 'Updating the inverse of a matrix', *SIAM Review* **31**(2), 221–239. **URL:** *http://www.jstor.org/stable/2030425*

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

Henderson, D. A., Boys, R. J., Krishnan, K. J., Lawless, C. and Wilkinson, D. J. (2009), 'Bayesian emulation and calibration of a stochastic computer model of mitochondrial DNA deletions in substantia nigra neurons', *Journal of the American Statistical Association* **104**(485).

Henderson, D., Boys, R. and Wilkinson, D. (2010), 'Bayesian calibration of a stochastic kinetic computer model using multiple data sources', *Biometrics* **66**(1), 249–256.

Hensman, J., Rattray, M. and Lawrence, N. D. (2014), 'Fast nonparametric clustering of structured time-series', *IEEE transactions on pattern analysis and machine intelligence* **37**(2), 383–393.

Hjort, N. L., Holmes, C. C., Müller, P. and Walker, S. G., eds (2010), *Bayesian Nonparametrics*, Cambridge University Press, Cambridge UK.

Ishwaran, H. and James, L. F. (2001), 'Gibbs sampling methods for stick-breaking priors', *Journal of the American Statistical Association* **96**(453), 161–173.

Ishwaran, H. and Zarepour, M. (2002), 'Exact and approximate sum representations for the Dirichlet process', *Canadian Journal of Statistics* **30**(2), 269–283.

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

Kloeden, P. E. and Platen, E. (2013), *Numerical solution of stochastic differential equations*, Vol. 23, Springer Science & Business Media.

Komorowski, M., Finkenstädt, B., Harper, C. V. and Rand, D. A. (2009), 'Bayesian inference of biochemical kinetic parameters using the linear noise approximation', *BMC bioinformatics* **10**(1), 343.

Kullback, S. and Leibler, R. A. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**(1), 79–86.

Kurtz, T. G. (1970), 'Solutions of ordinary differential equations as limits of pure jump Markov processes', *J. Appl. Probab.*. **7**, 49–58.

Kurtz, T. G. (1972), 'The relationship between stochastic and deterministic models for chemical reactions', *The Journal of Chemical Physics* **57**(7), 2976–2978.

Li, T., Sun, S., Sattar, T. P. and Corchado, J. M. (2014), 'Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches', *Expert Systems with applications* **41**(8), 3944–3954.

Liepe, J., Barnes, C., Cule, E., Erguler, K., Kirk, P., Toni, T. and Stumpf, M. P. (2010), 'ABC-SysBio—approximate Bayesian computation in Python with GPU support', *Bioinformatics* **26**(14), 1797–1799.

Liepe, J., Kirk, P., Filippi, S., Toni, T., Barnes, C. P. and Stumpf, M. P. (2014), 'A framework for parameter estimation and model selection from experimental data in systems biology using approximate Bayesian computation', *Nature Protocols* **9**(2), 439.

Lin, J. and Ludkovski, M. (2014), 'Sequential Bayesian inference in hidden Markov stochastic kinetic models with application to detection and response to seasonal epidemics', *Statistics and Computing* **24**(6), 1047–1062.

Liu, J. and West, M. (2001), Combined parameter and state estimation in simulation-based filtering, *in* 'Sequential Monte Carlo methods in practice', Springer, pp. 197–223.

Lotka, A. J. (1910), 'Contribution to the theory of periodic reactions', *The Journal of Physical Chemistry* **14**(3), 271–274.

MacEachern, S. N. and Müller, P. (1998), 'Estimating mixture of Dirichlet process models', *Journal of Computational and Graphical Statistics* **7**(2), 223–238.

Marjoram, P., Molitor, J., Plagnol, V. and Tavaré, S. (2003), 'Markov chain Monte Carlo without likelihoods', *Proceedings of the National Academy of Sciences* **100**(26), 15324–15328.

McDowell, I. C., Manandhar, D., Vockley, C. M., Schmid, A. K., Reddy, T. E. and Engelhardt, B. E. (2018), 'Clustering gene expression time series data using an infinite Gaussian process mixture model', *PLoS Computational Biology* **14**(1), e1005896.

McKay, M. D., Beckman, R. J. and Conover, W. J. (1979), 'Comparison of three methods for selecting values of input variables in the analysis of output from a computer code', *Technometrics* **21**(2), 239–245.

McKinley, T. J., Ross, J. V., Deardon, R. and Cook, A. R. (2014), 'Simulation-based Bayesian inference for epidemic models', *Computational Statistics & Data Analysis* **71**, 434–447.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953), 'Equation of state calculations by fast computing machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.

Morris, M. D. and Mitchell, T. J. (1995), 'Exploratory designs for computational experiments', *Journal of statistical planning and inference* **43**(3), 381–402.

Morzfeld, M., Fournier, A. and Hulot, G. (2017), 'Coarse predictions of dipole reversals by low-dimensional modeling and data assimilation', *Physics of the Earth and Planetary Interiors* **262**, 8–27.

Neal, R. M. (2000), 'Markov chain sampling methods for Dirichlet process mixture models', *Journal of Computational and Graphical Statistics* **9**(2), 249–265.

O'Hagan, A. (2006), 'Bayesian analysis of computer code outputs: A tutorial', *Reliability Engineering & System Safety* **91**(10-11), 1290–1300.

Owen, J., Wilkinson, D. J. and Gillespie, C. S. (2015), 'Likelihood free inference for Markov processes: a comparison', *Statistical applications in genetics and molecular biology* **14**(2), 189–209.

Papaspiliopoulos, O. and Roberts, G. O. (2008), 'Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models', *Biometrika* **95**(1), 169–186.

Purutcuoglu, V. and Wit, E. (2007), 'Bayesian inference of the kinetic parameters of a realistic MAPK/ERK pathway', *BMC Syst. Biol.* **1**, P19.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: *https://www.R-project.org/*

Raftery, A. E. and Lewis, S. M. (1992), 'Practical Markov Chain Monte Carlo: Comment: one long run with diagnostics: implementation strategies for Markov Chain Monte Carlo', *Statistical Science* **7**(4), 493–497.

Rasmussen, C. E. and Williams, C. K. I. (2005), *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*, The MIT Press.

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

Ross, J. and Dy, J. (2013), Nonparametric mixture of Gaussian processes with constraints, *in* 'International Conference on Machine Learning', pp. 1346–1354.

Rougier, J., Sexton, D. M., Murphy, J. M. and Stainforth, D. (2009), 'Analyzing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments', *Journal of Climate* **22**(13), 3540–3557.

Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. P. (1989), 'Design and analysis of computer experiments', *Statistical Science* pp. 409–423.

Schlögl, F. (1972), 'Chemical reaction models for non-equilibrium phase transitions', *Zeitschrift für physik* **253**(2), 147–161.

Schnoerr, D., Sanguinetti, G. and Grima, R. (2017), 'Approximation and inference methods for stochastic biochemical kinetics: a tutorial review', *Journal of Physics A: Mathematical and Theoretical* **50**(9), 093001.

Sethuraman, J. (1994), 'A constructive definition of Dirichlet priors', *Statistica sinica* pp. 639–650.

Smith, A. (2013), *Sequential Monte Carlo methods in practice*, Springer Science & Business Media.

Smith, A. F. and Gelfand, A. E. (1992), 'Bayesian statistics without tears: a sampling–resampling perspective', *The American Statistician* **46**(2), 84–88.

Sylvester, J. J. (1851), 'Xxxvii. on the relation between the minor determinants of linearly equivalent quadratic functions', *Philosophical Magazine Series 4* **1**(4), 295–305.

Teh, Y. W. (2011), Dirichlet process, *in* 'Encyclopedia of machine learning', Springer, pp. 280–287.

Tierney, L. (1994), 'Markov chains for exploring posterior distributions', *the Annals of Statistics* pp. 1701–1728.

Toni, T., Welch, D., Strelkowa, N., Ipsen, A. and Stumpf, M. P. (2008), 'Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems', *Journal of the Royal Society Interface* **6**(31), 187–202.

Van Kampen, N. G. (1992), *Stochastic Processes in Physics and Chemistry*, Vol. 1, Elsevier.

Vellela, M. and Qian, H. (2009), 'Stochastic dynamics and non-equilibrium thermodynamics of a bistable chemical system: the Schlögl model revisited', *Journal of The Royal Society Interface* **6**(39), 925–940.

Volterra, V. (1926), 'Fluctuations in the abundance of a species considered mathematically', *Nature* **118**, 196–218.

Wilkinson, D. J. (2018), *Stochastic modelling for systems biology*, 3rd edn, CRC Press.

Yule, G. U. et al. (1925), 'Ii.a mathematical theory of evolution, based on the conclusions of Dr. JC Willis, fr s', *Phil. Trans. R. Soc. Lond. B* **213**(402-410), 21–87.