Bayesian inference for stable differential equation models

with applications in computational neuroscience

Author:

Philip Maybank

A thesis presented for the degree of

DOCTOR OF PHILOSOPHY

SCHOOL OF MATHEMATICAL, PHYSICAL AND COMPUTATIONAL SCIENCES

UNIVERSITY OF READING

February 16, 2019

Remember to look up at the stars and not down at your feet.

Try to make sense of what you see and wonder about what makes the universe exist.

Be curious. And however difficult life may seem,

there is always something you can do and succeed at.

It matters that you don't just give up.

THE LATE PROF STEPHEN HAWKING, 1942-2018

ABSTRACT

Inference for mechanistic models is challenging because of nonlinear interactions between model parameters and a lack of identifiability. Here we focus on a specific class of mechanistic models, which we term stable differential equations. The dynamics in these models are approximately linear around a stable fixed point of the system. We exploit this property to develop fast approximate methods for posterior inference. We first illustrate our approach using simulated EEG data on the Liley *et al* model, a mechanistic neural population model. Then we apply our methods to experimental EEG data from rats to estimate how parameters in the Liley *et al* model vary with level of isoflurane anaesthesia. More generally, stable differential equation models and the corresponding inference methods are useful for analysis of stationary time-series data. Compared to the existing state-of-the art, our methods are several orders of magnitude faster, and are particularly suited to analysis of long time-series (>10,000 time-points) and models of moderate dimension (10-50 state variables and 10-50 parameters.)

ACKNOWLEDGEMENTS

I have enjoyed studying Bayesian methodology with Richard Culliford and Changqiong Wang, and Neuroscience with Asad Malik and Catriona Scrivener.

I have also been fortunate in being able to participate in the following graduate level training, and international conferences : the 2014-15 Academy for PhD Training in Statistics (APTS); the New Perspectives in Markov Chain Monte Carlo (NPMCMC) 2015 summer school; the International Society for Bayesian Analysis (ISBA) 2016 world meeting in Sardinia; and the Computational Neurosciences (CNS) 2018 meeting in Seattle. And further training in computational skills and software development at the Numerical Algorithms Group (NAG) in Oxford, 2017.

The support staff at the University of Reading (Ruth Harris, Brigitte Calderon, Peta-Ann King, and Kristine Aldridge) have given me a lot of practical help and general supportiveness.

My PhD supervisors, Richard Everitt and Ingo Bojak, have taught me a lot about their respective fields, and about myself.

I would like to thank my family, Gemma and Rowan, for their love and encouragement.

DECLARATION

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Philip Maybank

CONTENTS

CONTENTS (WITH SUBSECTIONS)

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

The problem addressed in this thesis is primarily one of scientific computing. Given a biophysical model of neural activity with unknown model parameters and time-series from the Electroencephalogram (EEG), how do you compute accurate estimates of the posterior distribution of the model parameters?

The work in this thesis is motivated by the following Neuroscience research question - what are the physiological mechanisms that cause the brain to lose consciousness during anaesthesia? A range of different hypotheses have been developed to answer this question. Our goal is to develop statistical methodology that makes it possible to determine which of these hypotheses best explains experimental observations. In the long term, research in this area may inform how anaesthetics are administered. Current clinical approaches to anaesthesia tend to over-anaesthetize. This increases the risk of side-effects such as post-operative delirium. Being able to determine the optimal doses of anaesthetic through a more detailed understanding of brain physiology would likely reduce these side-effects.

The areas of scientific computing most relevant to this thesis are Bayesian Computation and Computational Neuroscience. Early research in Bayesian Computation lead to the development of the Kalman filter for estimation and prediction in state-space models [Kalman, 1960], which was used for navigation systems in the Apollo space program in the 1960's. And early research in Computational Neuroscience lead to the development of the Hodgkin-Huxley equations as

a quantitative model for the nonlinear dynamics of ionic currents in nerve cells [Hodgkin and Huxley, 1952], which won a Nobel Prize in 1963.

In the intervening years the need for advances in scientific computing has not abated. This need is driven by increases in problem complexity as a result of larger datasets, larger numbers of model parameters, and more complex nonlinear models.

From the early 1990's onwards the field of Bayesian Computation grew rapidly as statisticians sought to address some of these challenges in a systematic and generic way, e.g. through the development of Markov Chain Monte Carlo (MCMC) [Gilks et al., 1995, Brooks et al., 2011].

In parallel to this, the field of Computational Neuroscience has also grown exponentially, as an increasingly wide range of computational models are developed to explain different aspects and levels of the nervous system [Dayan and Abbott, 2001, Izhikevich, 2007].

Developments in Bayesian Computation offer a powerful set of tools that can be applied to problems in Computational Neuroscience. However, developments in Computational Neuroscience also present statistical challenges that necessitate the development of novel statistical methodology. In both fields, and especially at the intersection, there is a pressing need for well-written software, so that Bayesian Computation methods can be tested and benchmarked on Computational Neuroscience models, and so that Computational Neuroscience practitioners can apply methods from Bayesian Computation without the need to invest excessive amounts of time in understanding how these methods work.

Dynamic Causal Modelling (DCM) is a set of Bayesian methods for combining models of brain physiology with macroscopic recordings of brain activity. The contributions of this thesis in relation to the existing DCM literature are as follows. The DCM community have recently applied Riemannian MCMC methods to sample parameters in ODE models [Penny and Sengupta, 2016]. I have extended the use of Riemannian MCMC to SDEs that have a stable fixed point (stable SDEs), also referred to as steady-state response models in the DCM literature [Moran et al., 2009]. The existing DCM literature on stable SDEs uses spectral analysis to define a likelihood

for time-series data that is transformed into the frequency domain. In Chapter 4, I apply an idea from the spectral analysis literature, called the Whittle likelihood, to stable SDEs. Whereas the likelihood used in spectral DCM uses an approximation that cannot be corrected or quantified, the Whittle likelihood is asymptotically exact for linear processes as the length of the time-series goes to infinity. The asymptotic error can be quantified using existing results from the spectral analysis literature, and the error that arises from linearizing around a stable fixed point can be quantified by comparing the Whittle likelihood with the particle filter likelihood, a more computationally expensive method that can be used to evaluate unbiased likelihood estimates for nonlinear dynamic models.

As well as accuracy of the model likelihood, a further issue in applying Bayesian Computation is scalability with the number of model parameters, and the length of the time-series. This issue becomes increasingly relevant as Computational Neuroscientists seek to fit more complex models to data so that a more detailed understanding of brain physiology can be obtained. I derived analytic derivatives for the Whittle likelihood, which are faster to calculate and more accurate than finite differences (Section 4.3.1). And I developed a novel reparameterization for stable SDE models (defined in Chapter 5), that reduces the complexity of the posterior distribution. Both these contributions lead to increased sampling efficiency of MCMC.

I implemented these methods in R and then subsequently in C++ (as described in Chapter 6). The R language is used by many researchers working in developing methods for Bayesian Computation, and offers an environment where it is possible to rapidly prototype and test novel methodology. C++ is much more commonly used than R in Computational Neuroscience. Existing implementations of Bayesian Computation methods tend to be part of packages, such as Stan [Carpenter et al., 2017] and SPM [Penny et al., 2011], where the details of the underlying algorithm are hidden from the user. I designed my C++ code to be more like a library than a package, which means that it should be possible for others who have their own code bases (either for modelling or for statistical methodology) to interface with my code more easily.

In Chapter 7, I then apply these methods to estimate physiological parameters from rat EEG data at multiple levels of the anaesthetic isoflurane. I used the Liley *et al* model, which previously modelled the effect of isoflurane solely through cortical parameters. I found that cortical effects were not sufficient for obtaining a good fit across multiple datasets, but that by additionally including the effect of isoflurane on the thalamic input, it was possible to obtain good fits to the data.

In discussing the contributions of Chapters 4 and 5 I emphasized that the novel methods that I developed are more accurate than existing DCM methods for spectral analysis of EEG data. A natural question is whether other existing methods from the wider statistical literature could have been applied to the problem of interest? Prior to developing the approach just outlined I explored whether particle MCMC [Andrieu et al., 2010] could be used. Particle MCMC can in principle be applied to any nonlinear state-space model and converges to the true posterior, whereas the method I developed converges to an approximation of the posterior. However I found that particle MCMC was too computationally intensive to be practical for the type of EEG data analysis problem we were interested in. This is because of the poor scalability of particle MCMC with the number of data-points, the number of unknown parameters, the degree of nonlinearity in the geometry of the posterior distribution and the dimension of the state-space, issues which we discuss in greater detail in Section 2.6.4. Following on from these early explorations, the logical next step was to seek methods that scaled better with these problem parameters. This lead me to draw heavily on existing statistical literature, bringing to together existing ideas in novel ways (for example using the Whittle likelihood within MCMC) and also developing novel methodological ideas (for example reparameterizing stable differential equations in terms of the equilibrium value).

As well as anaesthesia, the methods I have developed could also be applied in other application domains where time-series data is assumed to be stationary and estimation of mechanistic parameters is needed. For example, the spectral DCM methods discussed above have been ap-

plied in the context of Parkinson's Disease [Marreiros et al., 2013]. Applying the methods I have developed to this problem would likely lead to more accurate uncertainty quantification in this area. This is important if computational models are to be used to help direct the search for novel forms of Deep Brain Simulation (DBS) to treat Parkinson's Disease.

More generally methods for Bayesian analysis with mechanistic differential equation models have been developed for several other applications outside of Neuroscience. An early example in pharmacology is [Gelman et al., 1996] and a more recent example in muscle physiology is [Wheeler et al., 2014]. These studies do not make use of spectral analysis, the benefits of which are discussed in Chapter 4. As far as we are aware, spectral methods for analysis of mechanistic models are better developed within Neuroscience than in other application domains. In summary, the contributions of this thesis broaden the range of statistical methods that are available to statisticians both within and beyond Neuroscience.

2

SURVEY OF BAYESIAN METHODS

In this chapter we give an overview of tools that have been developed for Bayesian Computation. Many problems of interest in Bayesian inference involve high-dimensional integrals that cannot be evaluated exactly (or to within machine precision). The first class of methods we consider, approximate inference, replaces the integrals of interest with tractable integrals, such as a Gaussian density. These methods are typically fast, and in many applications are sufficient for the inference problem at hand. However, it can be difficult to quantify their accuracy. The second class of methods we discuss is Markov Chain Monte Carlo (MCMC). The basic idea of MCMC is to simulate from a Markov chain that has the posterior as its stationary distribution. MCMC is typically slower than approximate inference methods but comes with theoretical guarantees that integrals can be evaluated with any desired degree of accuracy, provided that the Markov chain is run for long enough. Some of the drawbacks of MCMC are that (i) if the target distribution is multi-modal the Markov chain can get stuck for a very long time in one mode, (ii) it is difficult to estimate marginal likelihoods using MCMC, which are important for some approaches to model comparison, (iii) the model needs to be written in a form such that the likelihood function can be evaluated. These limitations motivate the use of Sequential Monte Carlo (SMC), a very general class of methods that can overcome these issues, and also come with theoretical guarantees of convergence. These solutions come with a computational cost.

Some of the most powerful methods come from combining the basic methods above. Pseudo-marginal MCMC is an example of this where MCMC and SMC are combined. This has proven especially popular in state-space models, where a lot of work has been done on a type of SMC method called a particle filter. The way that the MCMC and SMC are combined means that an SMC algorithm must be executed on every iteration of an MCMC chain, leading some to describe these methods as 'computationally brutal', see discussion in [Andrieu et al., 2010]. This observation has lead to a renewed interest in approximate methods in the Bayesian Computation community, see for example [Chopin and Ridgway, 2017].

It is difficult to give general advice on how to tackle inference problems because they are so diverse. In my view approximate methods are useful for obtaining some initial results quickly. MCMC methods should be used to get more accurate results, and SMC methods should be used if multi-modality is a concern and/or if marginal likelihoods are needed. The optimal trade-off between accuracy and computational cost is often difficult to determine and problem dependent. Nevertheless it is a good idea to construct simplified inference problems where more accurate methods are tractable, in order to get some idea of the accuracy of approximate methods when applied to more challenging inference problems.

Within the neuroimaging literature, a popular approach, developed by the Dynamic Causal Modelling (DCM) community, is approximate inference that combines a variational approximation with the Laplace approximation. The DCM variational Laplace method has been validated / benchmarked against MCMC on a simple problem [Friston et al., 2007], and subsequently on a more complex problem [Penny and Sengupta, 2016]. In my PhD, I started by looking at whether particle MCMC could be used for parameter estimation in dynamic models for neuroimaging data. I got particle MCMC to work on a simple problem but found that the method did not scale well to the inference problem we were interested in. This lead me to develop approximate methods based on the Whittle likelihood, resulting in a computationally efficient method

that has important differences from existing methods, and has been benchmarked against more accurate methods on small problems, see Chapter 4.

It is also often possible to improve computational efficiency without sacrificing any accuracy by exploiting features that are model specific. Chapter 5 is an example of this for the case of differential equations with a stable steady-state.

2.1 PRELIMINARIES

Suppose that we have a dataset denoted by y, and that we have a model of the form $p_\theta(y|x)$, where x represents latent variables or state variables, θ represents model parameters or model hyper-parameters, and $p_\theta(y|x)$ is the likelihood of the parameters (θ, x) given some data y.

We are interested in the posterior distributions $p(\theta|y)$, obtained by calculating the marginal likelihood,

$$p(y|\theta) = \int p_\theta(y|x)p_\theta(x)\,dx, \tag{2.1}$$

and then applying Bayes' rule.

Furthermore, we may be interested in evaluating the model evidence, which is obtained by integrating over θ,

$$p(y|\mathcal{M}) = \int p(y|\theta)p(\theta)\,d\theta. \tag{2.2}$$

For the problems we are interested in, the marginal distribution $p(\theta|y)$ is of interest as the elements of θ are parameters of a physiological differential equation model. Assuming that the model represents the underlying data generating process in some meaningful and sufficiently accurate way, then inferring θ gives insight into the mechanisms that generate dynamics.

In the broader statistical and machine-learning literature θ may be regarded as a set of hyper-parameters. For example in a statistical model of spatial covariance, θ may represent the rate at which correlations in space decay with distance. Estimating hyperparameters from data is important for making better predictions and/or more accurately quantifying uncertainty in the parameters / latent variables in a wide range of models [Bishop, 2006, Rue et al., 2017].

2.1.1 *Maximum likelihood estimation*

Parameters can also be estimated using non-Bayesian or frequentist methods. Within statistics, perhaps the most common and basic method of estimation is to maximize the likelihood of the parameters. The result of this procedure is an estimator,

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ p(y|\theta). \tag{2.3}$$

In order to quantify uncertainty in the maximum likelihood framework, it is necessary to evaluate the sampling distribution of the estimator. Conceptually, the sampling distribution tells us how much variability we would see in our estimates if we were the repeat the data-generating process and re-calculate our estimate of θ.

Suppose that components of y are independent and identically distributed,

$$p(y|\theta) = \prod_{i=1}^{n} p(y_i|\theta), \tag{2.4}$$

and furthermore that θ^* does not lie on the boundary of the parameter space. Then, asymptotically, the estimator for θ is normally distributed.

We can approximate the sampling distribution of the estimator as,

$$\hat{\theta} \sim N(\theta^*, I(\theta^*)^{-1}) \quad \text{where} \quad I(\theta^*) = -\sum_{i=1}^{n} \frac{\partial^2 l(y_i;\theta^*)}{\partial\theta\partial\theta^T}, \tag{2.5}$$

where $l(y_i;\theta) = \log(p(y_i|\theta))$.

If the estimation problem is not close to the asymptotic case, it may be possible to find a more accurate parametric expression. Alternatively, a parametric or non-parametric bootstrap method can be used to estimate the sampling distribution. See, for example, Chapter 8 of [Hastie et al., 2009].

For big datasets where the posterior distribution is uni-modal and approximately Gaussian, the asymptotic sampling distribution is very similar to the posterior distribution. The distinctiveness of Bayesian methods compared with maximum likelihood methods is more marked when the

posterior distribution does not satisfy these conditions. For example, θ may only be weakly identifiable or non-identifiable. In this case, both the sampling distribution and the posterior will be non-Gaussian, even for large data-sets.

In cases where there is a good agreement between the sampling distribution of an estimator and the posterior, Bayesian methods can still be useful. For example they provide a principled approach to regularizing model parameters through the use of prior distributions. And the model evidence, $p(y|\mathcal{M})$, provides a way of comparing models that penalizes more complex models, again in a principled way.

2.1.2 *Optimization*

Optimization is needed to find maximum likelihood parameters, and is also useful or essential in many of the methods described below. For example, the number of iterations needed in an MCMC sampler can be reduced by initializing the sampler at the parameters that maximize the posterior density.

The most basic method of optimization is referred to as Newton's method, or Newton-Rhapson, see for example Chapter 13 of [Gelman et al., 2014]. In Newton's method a quadratic approximation to the objective function is constructed. On each iteration the quadratic approximation is maximized by applying the following update,

$$\theta_{n+1} = \theta_n - [\mathbf{H}f(\theta_n)]^{-1}\nabla f(\theta_n) \tag{2.6}$$

where $f(\cdot)$ is twice differentiable, $\nabla f(\theta_n)$ is a vector of partial derivatives at the point θ_n and $\mathbf{H}f(\theta_n)$ is the Hessian matrix, which contains the second partial derivatives of $f(\theta_n)$.

The algorithm stops when changes in the objective function pass below some user-defined threshold. If $-\mathbf{H}f(\theta_n)$, is not positive definite, the algorithm may not converge. Functions that are twice continuously differentiable and where $-\mathbf{H}f(\theta_n)$ is everywhere positive-definite are referred to as convex.

For non-convex functions, an alternative to the basic Newton method is to apply Newton's method component-wise, i.e., on each iteration all but one of the parameters is fixed. If $-\partial^2 f / \partial \theta_i^2 > 0$ then Newton's method is applied to the univariate function $f(\theta_i; \theta_{-i})$.

If the objective function is expensive to evaluate, is not smooth, and/or has many local optima then more complex optimization algorithms may need to be considered. It is outside the scope of this thesis to describe how these more complex algorithms are implemented. Documentation of optimization toolboxes and libraries, for example https://uk.mathworks.com/help/gads/... and https://www.nag.co.uk/.../e05intro.html provide a good overview of the most commonly used algorithms and what kind of problems they are most suited to.

One class of algorithms that is useful for problems in Chapter 7 of this thesis is Particle Swarm Optimization (PSO), [Kennedy and Eberhart, 1995, Mezura-Montes and Coello, 2011, Pedersen, 2010]. Unlike many other optimization algorithms, PSO does not require gradient evaluations, and is well suited to problems where the objective function is discontinuous and there are many local optima. Each iteration of PSO is relatively cheap, but PSO tends to require more iterations to converge than derivative-based methods. There are a number of user-defined parameters in PSO that can influence convergence, but knowing how to set these parameters can be a challenging. The convergence properties of PSO are not well understood from a mathematical perspective. It is sometimes heuristically argued that PSO is effective because it mimics swarm-like behaviour that is found in nature [Kennedy and Eberhart, 1995].

MCMC and SMC (described in more detail below) may also be used for the purposes of optimization. In the optimization algorithms considered thus far the primary outputs of the algorithm are the function arguments (e.g. parameter values) at the final iteration and function value (e.g. posterior density) at the final iteration. In MCMC and SMC the outputs of the algorithms are a set of parameter values that approximate the posterior distribution, and a set of posterior densities at these parameter values. So assuming that the samples generated by MCMC or SMC are a good approximation to the posterior density, the maximum of the

posterior density can be found by simply identifying the parameter set that had the highest posterior density. There are theoretical results (which are further discussed in Sections 2.3,2.4 showing the MCMC and SMC samplers converge to the posterior distribution. In practice there are many problems where MCMC and SMC have a tendency to get stuck in local mode(s). In these cases I would expect that restarting a deterministic optimization algorithm would be a quicker way of identifying additional modes. This question has not received much attention from either the optimization or computational statistics communities as far as I am aware. However a recent study in Computational Neuroscience argues that MCMC is more efficient than certain multistart optimization algorithms [Abeysuriya and Robinson, 2016].

The benefits of using MCMC and SMC over optimization are clearer when there is a need to quantify uncertainty in parameter values. Roughly speaking if we want to identify all parameter sets that result in a good fit to a dataset, then this information can be obtained using MCMC or SMC. In contrast, algorithms that are designed for efficient optimization can typically only be used to identify best fit parameters, or potentially the location of multiple local minima. Alternatively optimization algorithms may be used as a component within approximate inference algorithms to get partial uncertainty quantification at a relatively low computational cost.

2.2 APPROXIMATE INFERENCE

2.2.1 *The Laplace approximation*

The first step in the Laplace approximation is to find the mode of the posterior distribution,

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ p(\theta|y), \quad \text{where} \quad p(\theta|y) \propto p(y|\theta)p(\theta). \tag{2.7}$$

Then the posterior distribution is approximated by evaluating a second-order Taylor approximation of the log posterior density around the mode to obtain a Gaussian distribution,

$$\theta_q \sim N(\theta^*, \mathbf{H}(\theta^*)^{-1}) \quad \text{where} \quad \mathbf{H}(\theta^*) = -\frac{\partial^2 l_p(\theta)}{\partial\theta\partial\theta^T} - \sum_{i=1}^{n} \frac{\partial^2 l(y_i;\theta)}{\partial\theta\partial\theta^T} \tag{2.8}$$

where $l(y_i;\theta)$ is again $\log(p(y_i|\theta))$ and $l_p(\theta) = \log(p(\theta))$.

When model parameters are identifiable, the posterior distribution is asymptotically Gaussian, and so the Laplace approximation converges to the true posterior as $n \to \infty$, see Chapter 4 of [Gelman et al., 2014] for more details.

Although different in the interpretation attached to them, the Gaussian approximation to the sampling distribution and the Laplace approximation are identical when the prior distribution is uniform. The two distributions are also very close to each other when n is large since, in this case, the prior has a negligible effect on the Laplace posterior approximation.

It is straight-forward to approximate the marginal likelihood, $p(y)$, using the Laplace approximation,

$$p(y) \approx p(y|\theta^*)p(\theta^*)\sqrt{\frac{(2\pi)^d}{\det[\mathbf{H}(\theta^*)]}}, \tag{2.9}$$

where d is the number of components in θ.

2.2.2 *Variational approximation*

A variational approximation is a parametric probability density that minimizes the distance (typically the Kullback Leibler divergence) between some target density that we wish to approximate, and a family of probability densities of a specific form. Variational methods are used to simplify complex target distributions into a product of independent distributions. For example for the type of models considered in Section 2.1 we may consider approximations of the form,

$$q(\theta, x) = q(\theta)q(x). \tag{2.10}$$

Algorithm 1 gives a general form for optimizing the variational approximation in the equation above with respect to some target density of interest, $p(\theta, x)$. The Kullback Leibler (KL) divergence between the variational approximation and the posterior density can be written as,

$$\mathrm{KL}(q||p) = \mathrm{E}_{q(\theta,x)}\left[\log\left(\frac{p(\theta,x)}{q(\theta,x)}\right)\right] \tag{2.11}$$

$$= \mathrm{E}_{q(\theta)}\left[\mathrm{E}_{q(x)}\left[\log\left(p(\theta,x)\right)\right]\right] - \mathrm{E}_{q(\theta)}\left[\log\left(q(\theta)\right)\right] + \ldots \tag{2.12}$$

In the second line, we have kept only the terms that depend on $q(\theta)$. The KL divergence is minimized when the two distributions are identical. Assuming $q(x)$ is fixed, the minimum over $q(\theta)$ is attained when $\log(q(\theta)) = \mathrm{E}_{q(x)}[\log(p(\theta, x))]$, as specified in Algorithm 1. Algorithm 1 alternately updates $q(\theta)$ and $q(x)$ and is sometimes referred to as Coordinate Ascent Variational Inference (CAVI). More recently developed variational methods, sometimes referred to as Stochastic Variational Inference (SVI), estimate the gradient of the KL divergence by sub-sampling the data. SVI is currently a popular algorithm for big data problems where the full likelihood calculations required for CAVI are prohibitively costly. The interested reader is referred to the recent review in [Blei et al., 2017], and the references therein.

In some cases, for example mixture models with appropriate prior distributions, it is possible to derive analytical expressions for the variational density, without making any further approximations. The evaluation of expectations in Algorithm 1 has parallels with Expectation Maximization (or EM) algorithms. EM algorithms are used to find maximum likelihood parameter estimates, whereas variational Bayes can be used to approximate the posterior distribution for little additional computational cost. In addition to the review paper cited above, more details of how CAVI is implemented can be found in Chapter 10 of [Bishop, 2006].

Algorithm 1: Variational optimization

Input: initial setting θ; target distribution $p(\theta, x)$
Initialise $q(x) \propto p(x|\theta)$;
while KL *divergence has not converged* **do**
  Update $q(\theta)$ such that $\log q^*(\theta) = \mathrm{E}_{q(x)}[\log p(\theta, x)]$;
  Update $q(x)$ such that $\log q^*(x) = \mathrm{E}_{q(\theta)}[\log p(\theta, x)]$;
end

As discussed in Section 2.1, we are frequently interested in marginal likelihoods, for example integrating out x in a model parametrised by (θ, x) (Equation (2.1)), or integrating out all model parameters to obtain the model evidence, (Equation (2.2)).

Application of variational methods can result in approximate densities where it is possible to analytically calculate lower bounds for the marginal likelihood of interest. This enables an approximate form of model comparison.

In general, variational approximations are more accurate than the Laplace approximation because they do not necessarily assume that the target distribution is Gaussian, and they optimize with respect to a distance measure between probability distributions, and so are able to capture global properties of the target. In contrast the Laplace approximation optimizes the posterior density, and the resulting approximation is fully determined by the properties of the posterior at a single point in the parameter space.

CAVI is analogous to Gibbs sampling in that analytic calculations are required. In variational approximations these take the form of calculating expectations under $q(\theta)$ and $q(x)$. This can be a time-consuming process from a user point of view. And in general, it may not be possible to obtain analytic expressions for the required expectations. In this case, the components of the variational approximation can themselves be approximated, e.g., using the Laplace approximation [Friston et al., 2007]. For some problems this has been found to work better than applying the Laplace approximation to the original target density. For example, suppose that the target density contained two peaks, and that one peak had a higher maximum density, but the other peak contained most of the probability. A Laplace approximation of this target would put most of the probability around the highest peak. Because variational approximations optimize with respect to a distance between probability distributions, variational approximation combined with the Laplace approximation would put most of the probability in the variational density around the peak containing most of the probability in the posterior density.

A drawback of variational methods compared to MCMC is that they tend to under-estimate uncertainty in the posterior distribution as a result of using the Kullback-Leibler divergence as a measure of similarity between probability distributions.

For a more detailed discussion of variational methods see Chapter 13 of [Gelman et al., 2014] and Chapter 10 of [Bishop, 2006].

2.2.3 *Integrated Nested Laplace Approximation*

For some models there is good reason to think that the joint distribution, $p(\theta, x|y)$ is not Gaussian and cannot be well approximated by the Laplace approximation, but that the conditional distribution $p_\theta(x|y)$ is approximately Gaussian. Applying the Laplace approximation to the conditional enables us to approximate the marginal likelihood in Equation (2.1). If, in addition to the conditional posterior being approximately Gaussian, θ is sufficiently low dimensional that it is possible to numerically integrate functions of θ, we can also obtain posterior marginals for x, i.e.,

$$p(x_i|y) = \int p_\theta(x_i|y)p(\theta|y)d\theta \tag{2.13}$$

The method that is built on these ideas is called the Integrated Nested Laplace Approximation (INLA). For Gaussian Markov Random Fields, the integral over x that is required to evaluate $p(\theta|y)$ can be done quickly, which has enabled the method to be applied to spatial problems that have high dimensional latent variables [Lindgren et al., 2011]. In the original method, the integration over θ was done by evaluating the integrand on a grid of points, but as the method has developed, more sophisticated numerical integration methods have been used, which increase the dimension of θ for which the integral in Equation (2.13) is tractable up to around 10-20. See [Rue et al., 2017] for a recent review.

For the problems we are interested in the dimension of θ may be greater than 20, so our preferred method for estimating the posterior distribution $p(\theta|y)$ is to use Markov Chain Monte Carlo, which we introduce in the next section.

2.3 MARKOV CHAIN MONTE CARLO

2.3.1 *The Metropolis-Hastings algorithm*

In many cases of practical interest it is difficult to know how close the Laplace approximation is to the true posterior. The Metropolis-Hastings algorithm (Algorithm 2) can generate samples that are arbitrarily close to samples from the true posterior distribution, provided that the algorithm is run for a sufficiently long time. The efficiency with which these samples can be generated has been the subject of intensive research for the last 20-30 years. See [Green et al., 2015] for a recent overview.

One limitation of MCMC methods such as the MH algorithm is that they cannot be used directly to estimate marginal likelihoods. However, they can be used as a component within more sophisticated algorithms, such as Annealed Importance Sampling (AIS), which are used to estimate marginal likelihoods (see Section 2.4).

Algorithm 2: Metropolis-Hastings (MH)

Input: initial setting x, number of iterations I;
target distribution $\pi(x)$; proposal distribution $q(x^*|x)$.

for $i = 1, \ldots, I$ **do**

Propose $x^* \sim q(x^*|x)$;

Compute $\alpha(x, x^*) = \min\left[1, \frac{\pi(x^*)q(x|x^*)}{\pi(x)q(x^*|x)}\right]$;

Draw $u \sim \text{Uniform}(0, 1)$;

if $u < \alpha(x, x^*)$ **then** set $x = x^*$;

end

Sampling from posterior distributions is an application of the Metropolis-Hastings algorithm. Much of the Markov Chain Monte Carlo (MCMC) literature uses terminology that describes the algorithm in a more general framework. The probability distribution that we wish to sample from, $\pi(x)$, is referred to as the target distribution, and the distribution used to propose new x values, $q(x)$, is referred to as the proposal distribution. In continuous state-spaces, these distributions can be referred to as the target density and the proposal density.

There are several different classes of Metropolis-Hastings algorithm. Random Walk Metropolis Hastings (RWMH) does not use any local gradient information in the proposal distributions.

The Metropolis Adjusted Langevin Algorithm (MALA) and Hamiltonian Monte Carlo (HMC) require the gradient of the target density to be evaluated on each iteration. This local geometric information informs the proposal density, leading to more efficient samplers, and better scaling with the dimension of the target density. This is discussed further in Section 2.3.3.

The RWMH can also incorporate geometric information into the proposal density. This information tends to be global rather than local, and is typically computed in preliminary MCMC runs, or with an approximate inference method. For example, if the target distribution is approximately Gaussian, the Laplace approximation can be used to construct an effective proposal distribution as follows,

$$q(x^*|x) = N(x, c^2\hat{\Sigma}), \tag{2.14}$$

where $\hat{\Sigma}$ is the covariance matrix in the Laplace approximation, and $c = 2.4/\sqrt{d}$ is a constant, with d being the dimension of the target. The value of c is derived from theoretical work on optimizing MCMC sampling efficiency for a Gaussian target distribution. See Section 2.3.3 for further discussion.

An advantage of RWMH is that it is relatively easy to implement. Until recently, MALA and HMC were often challenging to implement effectively as they contain several user-defined parameters. Some guidance on setting these parameters can be found in [Neal, 2011], but in general this task is not straight-forward. More recently, use of HMC has become more widespread as a result of the No U-Turn Sampler (NUTS) which requires less manual tuning and is implemented in the Stan software [Hoffman and Gelman, 2014, Carpenter et al., 2017]. Another innovation that has resulted in easier to implement gradient-based MCMC has been the use of Riemannian geometry, which we discuss in Section 2.3.4.

2.3.2 *The Metropolis-within-Gibbs algorithm*

The Metropolis-within-Gibbs (MwG) algorithm (Algorithm 3) is a special case of the MH algorithm where the parameters are updated one at a time, or in blocks, rather than all at once.

Algorithm 3: Metropolis-within-Gibbs (MwG)

Input: initial setting x, number of iterations I;
target distribution $\pi(x)$;
proposal distributions $q_1(x_1^*|x_1), \ldots, q_p(x_p^*|x_p)$.

for $i = 2, \ldots, I$ **do**
  for $j = 1, \ldots, p$ **do**
    Set $x_{-j}^* = x_{-j}$;
    Propose $x_j^* \sim q_j(x_j^*|x_{1:p})$;
    Evaluate acceptance ratio, $\alpha(x, x^*) = \min\left[1, \frac{\pi(x^*)q_j(x_j|x_{1:p}^*)}{\pi(x)q_j(x_j^*|x_{1:p})}\right]$;
    Draw $u \sim \text{Uniform}(0, 1)$;
    if $u < \alpha(x, x^*)$ **then** Set $x = x^*$;
  end
end

The algorithm reduces to Gibbs sampling when the proposal densities are conditionals of the target density,

$$q_j(x_j^*|x_{-j}) = \pi(x_j|x_{-j}), \tag{2.15}$$

where $\pi(x_j|x_{-j})$ is distribution of x_j conditional on x_{-j}. The notation x_{-j} refers to elements of x that are not components of x_j.

The advantages of Gibbs sampling are that there are no parameters that need to be tuned, and it can be efficient even when the target is highly non-Gaussian. However, the requirement to derive conditional densities restricts the range of models that it can be applied to.

Another choice of proposal density which can be useful when analytical conditional densities are not available, is a univariate Gaussian proposal on blocks of size one,

$$q_j(x_j^*|x_j) = N(x_j, \sigma_j^2). \tag{2.16}$$

For higher-dimensional problems and non-Gaussian target distributions, MwG algorithms generally perform better than RWMH because it is easier to find a good proposal distribution in low dimensions than it is in higher dimensions. However, MwG will be inefficient when there are strong dependencies between x components in different blocks.

2.3.3 *Sampling efficiency and tuning MH and MwG algorithms*

MCMC sampling can be separated into two phases: the warm-up or burn-in period and the stationary period. During the warm-up period the sampler typically starts from a region of low probability density and moves towards a region of high probability density. Samples from the burn-in period are discarded for the purposes of estimation. Alternatively, the sampler can be initialized from a region of high posterior density, in which case no burn-in phase is required.

Ideally we would like each sample that we generate to be an independent draw from the probability distribution of interest. However MCMC always generates samples that are correlated with each other, to a greater or lesser extent. For example in the Metropolis-Hastings algorithm (Algorithm 2) we could choose a proposal distribution $q(x^*|x)$ with a small variance. In that case the sampler will have a high acceptance rate. The correlation between samples will be high because, on average, the sampler moves a small distance on each iteration. If we choose $q(x^*|x)$ with a large variance the sampler will have a low acceptance rate. The correlation between samples will be high because there is a high probability of not moving at all. It is therefore important to choose the parameters of an MCMC sampler is such a way that minimizes correlation thus maximizing sampling efficiency. Higher sampling efficiency means that fewer MCMC iterations are needed in order to accurately estimate expectations with respect to the target density.

In this section we focus on sampling efficiency during the stationary period of an MCMC algorithm and primarily consider the Effective Sample Size (ESS) as a measure of sampling efficiency. If we are sampling from a multivariate target distribution, the estimated ESS of the ith component is,

$$\text{ESS}_i = \frac{S}{1 + 2\sum_k \hat{\rho}_i(k)}, \tag{2.17}$$

where S is the number of samples obtained during the stationary period, and $\hat{\rho}_i(k)$, is an estimate of the autocorrelation at lag k for the ith component of the samples. This expression can be derived from writing down the variance of the average of a correlated sequence (Chapter 11 of

[Gelman et al., 2014]). The key point to note is that if the autocorrelation decays slowly the ESS will be relatively small.

As referred to in Section 2.3.1, the optimal proposal density for Random Walk MH is $c^2\Sigma$ where $c = 2.4/\sqrt{d}$ when the target is Gaussian with covariance Σ. The proposal density is optimal in the sense that it maximises the ESS. When the target is Gaussian, the ESS for RWMH with the optimal proposal density is approximately $0.3S/d$, [Roberts and Rosenthal, 2001]. Although it is difficult to make statements that apply to general target distributions, the theoretical result we have just stated suggests that S should increase linearly with d in order to maintain the ESS for RWMH.

Theoretical results for MALA, again on a Gaussian target, state that the ESS is $O(I/d^{1/3})$. This result supports the idea that MALA scales better with the dimension of the target distribution than RWMH. For example the theoretical results suggest that if $d = 50$, RWMH should require around 10 times as many iterations as MALA to obtain the same ESS. Whether the increased sampling efficiency outweighs the increased cost per iteration of MALA is problem-dependent. See [Roberts and Rosenthal, 2001] for an overview of results on optimal scaling.

The proposal density suggested in Equation (2.14) estimates Σ using the Laplace approximation. This can be improved by first running an MCMC sampler using Equation (2.14), and then estimating the Σ from the MCMC samples. Then a new run using the re-estimated covariance can be performed. Provided that the first MCMC run is sufficiently long, and that the Laplace approximation is not exact, the covariance estimates obtained from sampling the target distribution will be more accurate than the covariance estimate from the Laplace approximation.

Proposal variances for MwG with the proposal from Equation (2.16) can be tuned by doing a pilot run, for example using the marginal variances from the Laplace approximation. Then the conditional variance of the ith parameter can be estimated by fitting a regression model with the ith parameter as the response variable, and all other parameters as explanatory variables. See [Raferty and Lewis, 1996] for more details.

Another approach to setting parameters of the proposal distribution is to use adaptive MCMC [Andrieu and Thoms, 2008]. Adaptive MCMC offers a potentially greater degree of automation than the methods described above. However, adaptive MCMC methods are more challenging to implement effectively.

2.3.4 *Riemannian MCMC*

The basic idea of Riemannian MCMC is to use higher-order derivatives of the target density to design better proposals. Riemannian methods also reduce the number of user-defined parameters making them relatively easy to implement. For example, the simplified manifold MALA (smMALA) uses the first and second order derivatives - Algorithm 4. In general, these are $O(d)$ and $O(d^2)$ calculations, which can make the smMALA algorithm quite computationally expensive. However, the use of local geometry on each iteration means that the efficiency per MCMC iteration is very high (see Section 2.3.3 for more discussion). Empirically we have found for complex models with nonlinear dependencies, the costly derivative calculations are outweighed by the benefits of a better proposal.

The fully Riemannian approach uses third order derivatives of the target density. This introduces an extra factor of d at each MCMC iteration making the method prohibitively costly for high dimensional parameter spaces. For more details see [Girolami and Calderhead, 2011].

2.4 SEQUENTIAL MONTE CARLO

Both the Laplace approximation and variational approximations produce biased estimates of the marginal likelihood. Sequential Monte Carlo (SMC) is a family of algorithms that allow for unbiased estimation of the marginal likelihood. Whereas MCMC algorithms evolve a single Markov chain targeting some distribution of interest, SMC algorithms evolve a population of N particles and target a sequence of distributions, $\{\pi_0(x), \ldots, \pi_n(x)\}$. As $N \to \infty$, the variance of SMC estimates goes to zero. However, N may need to be very large in order to obtain sufficiently low-variance estimates.

Algorithm 4: Simplified manifold Metropolis Adjusted Langevin Algorithm (sm-MALA)

Input: Data, y; Initial value for $\theta = (\theta_1, \ldots, \theta_p)$;
Likelihood $l(y|\theta)$; Prior distribution $p(\theta)$.
Parameters: Step size, h; Number of iterations, I
for $i = 2, \ldots, I$ **do**
Evaluate gradient and Hessian of the unnormalized log posterior,
$g = \nabla \left[\log[\ l(y|\theta)\ p(\theta)\ ] \right]$ and $\mathbf{H}_\theta$;
Set $C = h^2 \mathbf{H}_\theta^{-1}$, and $m = \theta + \frac{1}{2} C\, g$;
Propose $\theta^* \sim q(\theta^*|\theta) = N(m, C)$;
Evaluate acceptance ratio, $\alpha(\theta, \theta^*) = \min \left[1, \frac{l(y|\theta^*)p(\theta^*)q(\theta|\theta^*)}{l(y|\theta)p(\theta)q(\theta^*|\theta)} \right]$;
Draw $u \sim \text{Uniform}(0, 1)$;
if $u < \alpha(\theta, \theta^*)$ **then** Set $\theta = \theta^*$;
end

SMC algorithms have their origins in inference for dynamical systems where observations arrive sequentially in time. This literature is reviewed in [Doucet et al., 2001]. The first presentation of a general SMC framework was [Del Moral et al., 2006] and in recent years there has been an increasing use of SMC for a range of static problems [Jasra et al., 2007, Everitt, 2012, Kantas et al., 2014, Zhou et al., 2016]. We present a general SMC algorithm in Algorithm 5. In the text that follows we highlight several important classes of SMC algorithm.

We first make a couple of remarks on Algorithm 5. The SMC literature tends to use the term kernel to describe probability distributions that play an analogous role to proposal distributions in MCMC. This is discussed further in the context of specific examples below. Algorithm 5 contains the expression $\text{ESS} = \left(\sum_{i=1}^N w_j^{(i)}\right)^2 / \sum_{i=1}^N \left(w_j^{(i)}\right)^2$. As in Section 2.3.3, the abbreviation ESS stands for Effective Sample Size. Whereas in MCMC, the ESS is reduced by autocorrelation between successive MCMC samples, in SMC the ESS is reduced by variance in the particle weights. If most of the particles have low weights and a small number of particles have relatively high weights, the ESS will be low. This phenomenon is referred to as sample degeneracy. In the limiting case all of the weight could be on a single particle, in which case the ESS is equal to 1. If all of the weights are equal, the ESS is equal to N, the number of particles. Recently it has been argued that the ESS expression used in Algorithm 5 cannot be considered a reasonable ap-

proximation to the underlying ESS [Elvira et al., 2018]. However it continues to be a commonly used diagnostic for assessing sample degeneracy in the context of SMC.

When $n = 1$, SMC reduces to importance sampling. Let $\pi_0(x)$ be a normalised proposal distribution, and $\pi_1(x)$ be some unnormalized density, i.e., $p_1(x) = \pi_1(x)/Z$. Then if $w^{(i)} = \pi_1(x^{(i)})/\pi_0(x^{(i)})$, with $x^{(i)} \sim \pi_0(x)$,

$$\mathrm{E}_{\pi_0}[w^{(i)}] = \int w(x)\pi_0(x)dx = Z. \tag{2.18}$$

The quantity $\frac{1}{N}\sum_{i=1}^{N} w^{(i)}$ is therefore an unbiased importance sampling estimate of Z. For high-dimensional targets, importance sampling estimates have a very high variance, and more sophisticated SMC algorithms are needed. In particular resampling through $P(x_{j-1}^{(1)}, \ldots, x_{j-1}^{(N)}; w_j^{(1)}, \ldots, w_j^{(N)})$ using one of the methods in Section 2.4.1 can reduce the variance.

Algorithm 5: Sequential Monte Carlo (SMC)

Input: threshold for triggering resampling, S;
sequence of target distributions, $\pi_1(x_1), \ldots, \pi_n(x_{1:n})$;
initial proposal distribution, $q(x_1)$;
sequence of proposal distributions / kernels, $K_2(x_1, x_2), \ldots, K_n(x_{n-1}, x_n)$.

for $i = 1, \ldots, N$ **do**
- Propose, $x_1^{(i)} \sim q(x_1)$
- Weight, $w_1^{(i)} = \dfrac{\pi_1(x_1^{(i)})}{q(x_1^{(i)})}$

end
for $j = 2, \ldots, n$ **do**
- **if** ESS $= \left(\sum_{i=1}^{N} w_j^{(i)}\right)^2 / \sum_{i=1}^{N} \left(w_j^{(i)}\right)^2 < S$ **then**
  - Resample, $x_{j-1}^{(i)} \sim P(x_{j-1}^{(1)}, \ldots, x_{j-1}^{(N)}; w_j^{(1)}, \ldots, w_j^{(N)})$
- **end**
- **for** $i = 1, \ldots, N$ **do**
  - Propagate / Mutate, $x_j^{(i)} \sim K_j(x_{j-1}^{(i)}, x_j)$
  - Reweight, $w_j^{(i)} = \dfrac{\pi_j(x_{1:j}^{(i)})}{\pi_{j-1}(x_{1:(j-1)}^{(i)}) K_j(x_{j-1}^{(i)}, x_j^{(i)})} \cdot w_{j-1}^{(i)}$
- **end**

end
if ESS $= \left(\sum_{i=1}^{N} w_n^{(i)}\right)^2 / \sum_{i=1}^{N} \left(w_n^{(i)}\right)^2 < S$ **then**
- Resample, $x_j^{(i)} \sim P(x_j^{(1)}, \ldots, x_j^{(N)}; w_j^{(1)}, \ldots, w_j^{(N)})$

end

SMC algorithms for inference in state-space models are often referred to as particle filters. The sequence of target distributions for a state-space model is the set of joint distributions,

$$\pi_n(x_{1:n}) = p_\theta(x_{1:n}, y_{1:n}). \tag{2.19}$$

for $n \geq 1$, [Doucet and Johansen, 2009].

In the bootstrap particle filter, the kernel K is chosen to be the prior dynamics of the model conditioned on the previous state, $p_\theta(x_n|x_{n-1})$. We discuss particle filters in more detail in Section 2.6.3.

Another example of an SMC algorithm is Annealed Importance Sampling (AIS), which is useful for estimating ratios of marginal, or otherwise intractable, likelihoods in cases where efficient MCMC sampling from $\pi_n(x)$ is possible [Neal, 2001].

Given a pair of unnormalized target densities, $\pi_0(x)$ and $\pi_n(x)$, for which we would like to estimate the normalising constant ratio, Z_n/Z_0, we define our sequence of target distributions as follows,

$$\pi_j(x) = \pi_0(x)^{\gamma_j}\ \pi_n(x)^{(1-\gamma_j)}, \tag{2.20}$$

where $\gamma_0 = 0$, $\gamma_n = 1$ and $(\gamma_1, \ldots \gamma_{n-1}) \in (0, 1)$, with $\gamma_j < \gamma_{j+1}$ for $j = 0, \ldots, n-1$. The distributions $\pi_1(x), \ldots, \pi_{n-1}(x)$ are often referred to as intermediate distributions.

AIS can be done with or without resampling. No resampling corresponds to setting P to a δ function centred on the current particle location, $\delta_{x_{j-1}^{(i)}}(x)$. When there is no resampling, no communication is required between particles, and the algorithm is trivially parallelizable. This makes it better suited to cluster architectures where there are thousands of processors spread across multiple nodes, and communication between nodes is relatively slow. The drawback of not resampling is that it can become difficult to control the variance of the weights. An adaptive resampling scheme, where resampling is triggered when the ESS falls below a certain threshold, is less likely to suffer from degeneracy issues [Jasra et al., 2007].

To obtain the AIS algorithm from the general SMC formulation in Algorithm 5, we choose the kernel K_j to be an MCMC kernel that targets $\pi_j(x)$.

We define a sequence of auxiliary target distributions, $\tilde{\pi}_n(x_{0:n})$,

$$\tilde{\pi}_n(x_{0:n}) = \pi_n(x_n) \prod_{j=2}^{n} L_{j-1}(x_j, x_{j-1}) \tag{2.21}$$

with,

$$L_{j-1}(x_j, x_{j-1}) = \frac{\pi_j(x_{j-1}) K_j(x_{j-1}, x_j)}{\pi_j(x_j)}. \tag{2.22}$$

where L is a *backward* in time Markov kernel.

The auxiliary distribution $\tilde{\pi}_n(x_{0:n})$ admits $\pi_n(x_n)$ as a marginal distribution. So we can generate samples from $\pi_n(x_n)$ by sampling from $\tilde{\pi}_n(x_{0:n})$

For these choices of K and L kernels, the weights can be written as,

$$w_n^{(i)} = \prod_{j=1}^{n} \frac{\pi_j(x_{j-1}^{(i)})}{\pi_{j-1}(x_{j-1}^{(i)})}. \tag{2.23}$$

The expectation of $w_n^{(i)}$ is Z_n / Z_0, so $\frac{1}{N} \sum_{i=1}^{N} w_n^{(i)}$ will be a good estimate for Z_n / Z_0 when N is large. Increasing n, which controls the number of intermediate distributions, also reduces the variance of Z_n / Z_0.

The choice of L in Equation (2.22) is specific to the AIS algorithm. However the definition of the (auxiliary) target distributions in Equation (2.21) is general to all SMC algorithms. SMC users are free to define L to be any probability distribution for sampling x_{j-1} given x_j. Ideally we would choose L in such a way as to minimize the variance of the importance weights. However the optimal L often leads to weight expressions that are intractable. See Section 3.3 of [Jasra et al., 2007] for more details.

2.4.1 *Resampling distributions*

The most popular choices of resampling algorithms are,

- Multinomial. Sample from a discrete distribution with support on the current particle values. The parameters of the distribution are the normalised weights.

- Systematic. Sample U_1 from a uniform distribution on the interval $(0, 1/N)$. Define $U_i = U_1 + \frac{i-1}{N}$ for $i = 2, \ldots, N$. Then calculate

$$N_t^i = \left| \left\{ U_j : \sum_{k=1}^{i-1} W_t^k \leq U_j \leq \sum_{k=1}^{i} W_t^k \right\} \right|.$$

- Residual. Let $R = \sum_{i=1}^{N} \lfloor N w_t^i \rfloor$ and $N_t^i = \lfloor N w_t^i \rfloor + \tilde{N}^i$, with $\tilde{N}^i$ sampled according to a multinomial distribution. The multinomial parameters are set to $\tilde{w}^i = \frac{N w_t^i - \lfloor N w_t^i \rfloor}{N-R}$, and the distribution is sampled $N - R$ times.

Systematic and Residual resampling have a lower variance than multinomial and therefore result in lower variance particle filter estimates than multinomial resampling.

2.5 PSEUDO-MARGINAL MCMC

We may wish to implement a marginal Metropolis-Hastings algorithm, where on each iteration we evaluate the marginal likelihood $p(\theta|y)$, where x has been integrated out. For some models the x variables are not of interest, and when they are integrated out the MCMC sampler is more efficient. There are many cases where the x variables cannot be integrated out analytically, but we can obtain a Monte Carlo estimate of the integral using SMC.

The idea of pseudo-marginal MCMC is to use the Monte Carlo estimate in place of the true marginal likelihood. It can be shown that this "noisy" algorithm targets the marginal distribution of θ [Andrieu and Roberts, 2009, Andrieu and Vihola, 2015].

Before pseudo-marginal algorithms were discovered, the only way to sample from the marginal of θ was to sample from the joint distribution (θ, x). Pseudo-marginal algorithms offer an alternative to this approach, and for many problems it has been easier to design SMC samplers to estimate marginal likelihoods than to design MCMC samplers that target the joint distribution of (θ, x). We discuss the case of using particle filters to estimate marginal likelihoods in nonlinear state-space models in 2.6.4. Other examples include using importance sampling to estimate the marginal likelihood of parameters in statistical genetics models [Beaumont, 2003], estimating the partition function in Ising models in order to sample the Ising model parameters [Everitt,

2012], and using AIS within reversible jump MCMC for sampling the posterior distribution over the number of components in a mixture model [Karagiannis and Andrieu, 2013].

2.6 STATE-SPACE MODELS

We start this section by defining linear and nonlinear dynamical systems in discrete-time. The dynamical systems we consider are noisy in the sense that random variables influence the evolution of the dynamical system. The following subsections discuss Bayesian methods for estimation in state-space models. A state-space model is simply a noisy dynamical system for the evolution of latent states coupled to a statistical model for observations given the latent state variables.

In the presentation of estimation methods (Sections 2.6.2-2.6.4) we primarily consider the discrete-time case as this is simpler from a technical point of view, and more relevant to numerical implementations. However many physical models are formulated in continuous-time. We will therefore also briefly consider methods for discretizing noisy continuous-time systems, i.e., Stochastic Differential Equations (SDEs) in Section 2.6.1.

A linear-Gaussian discrete-time dynamical system can be written as,

$$\mathbf{X}_{j+1} = A\,\mathbf{X}_j + \mathbf{W}_j, \tag{2.24}$$

where $\mathbf{X}_1 \sim \mathrm{N}(\mu_0, P_0)$ and $\mathbf{W}_j \sim \mathrm{N}(0, Q)$ for $j \geq 1$ are d_x dimensional multivariate Gaussian random variables. The $d_x \times d_x$ matrix A defines the deterministic component of the dynamics, and the random variable $\mathbf{W}_j$ is the noise at time j.

More generally a nonlinear discrete-time dynamical system can be written as,

$$\mathbf{X}_{j+1} = \boldsymbol{\Phi}[\mathbf{X}_j, \mathbf{W}_j], \tag{2.25}$$

where $\mathbf{X}_1$ and $\mathbf{W}_j$ are now generic multivariate random variables, and $\boldsymbol{\Phi}$ is a function with d_x outputs. The dimension of $\mathbf{W}_j$ is not fixed, and in general may be larger than or smaller than $\mathbf{X}_j$.

The noise in a nonlinear dynamical system can be additive, in which case,

$$\mathbf{X}_{j+1} = \mathbf{\Phi}[\mathbf{X}_j] + \mathbf{W}_j, \tag{2.26}$$

where in this case $\mathbf{W}_j$ has dimension d_x.

In a linear-Gaussian state-space model, the statistical model for observations is,

$$\mathbf{Y}_j = H\,\mathbf{X}_j + \mathbf{U}_j, \tag{2.27}$$

where $\mathbf{U}_j \sim \mathbf{N}(0, \Sigma)$ is a d_y dimensional multivariate Gaussian random variable that represents noise in measurements. The $d_y \times d_x$ matrix H defines the expected value of observations given the state variables. Bayesian estimation for linear-Gaussian state-space models can be done using the basic Kalman filter, which we discuss in Section 2.6.2.

In a nonlinear state-space model, the statistical model for observations is,

$$\mathbf{Y}_j = \mathbf{h}(\mathbf{X}_j, \mathbf{U}_j) \tag{2.28}$$

where $\mathbf{U}_j$ is now a generic multivariate random variable. Bayesian estimation for general nonlinear state-space models can be done using the bootstrap particle filter, [Gordon et al., 1993]. When the process noise is additive (as in Equation (2.26)) estimation can be done using extensions of the basic Kalman filter, such as the Extended Kalman Filter (EKF), or the Unscented Kalman Filter (UKF), see [Anderson and Moore, 1979], [Grewal and Andrews, 1993], and [Julier and Uhlmann, 1997]. Alternatively, the EKF or UKF can be used within a particle filter, see for example the Unscented Particle Filter (UPF) in Section 2.6.3. The UPF was introduced in [Van Der Merwe et al., 2001] and this paper forms the basis for our presentation of the methods in Section 2.6.3.

The bootstrap particle filter is the most generally applicable of the algorithms we present for estimation in state-space models. It is an instance of an SMC algorithm and therefore produces unbiased Monte Carlo estimates. However, as we will see in Section 2.6.3, the computational cost of the bootstrap particle filter can be prohibitively high. For linear-Gaussian state space models

it is always preferable to use the basic Kalman filter because it is deterministic, unbiased and more computationally efficient. Extensions of the Kalman filter for nonlinear state-space models produce biased estimates, but they retain the benefits of being deterministic and relatively fast. Extensions of the bootstrap particle filter, e.g., the UPF, are typically more computationally efficient than the bootstrap particle filter but less generally applicable. The UPF is less computationally efficient than the UKF, but as an SMC algorithm, it produces unbiased Monte Carlo estimates.

For some problems, e.g. discretizations of spatial dynamical systems (e.g. PDEs), the computational cost of Kalman filters such as the EKF and UKF becomes prohibitive due to the high dimension of the state space. As will be discussed in Section 2.6.2, each iteration of a Kalman filter involves an $O({d_x}^3)$ matrix operation. The ensemble Kalman filter [Evensen, 1994] is a Monte Carlo variant of the Kalman filter that scales better with the dimension of the state space than the EKF and UKF. Unlike particle filters it is not an SMC algorithm so produces biased estimates. We do not further discuss the ensemble Kalman filter in this thesis, as we are primarily interested in problems where the state space has moderate dimension, e.g., 10-50 state variables.

2.6.1 *Discretization of continuous-time state-space models*

A linear-Gaussian continous-time dynamical system is usually written as,

$$d\mathbf{X}(t) = A\,\mathbf{X}(t)dt + d\mathbf{W}(t) \tag{2.29}$$

where $\mathbf{X}(0) \sim \mathrm{N}(\mu_0, P_0)$ and $\mathbf{W}(t) \sim \mathrm{N}(0, Q)$ is a Wiener process with d_x independent components.

Suppose that we directly observe the continuous-time system at uniformly spaced discrete time-points, $t = j \cdot \Delta t$ for $j \geq 0$. This discrete-time process is approximately equivalent to the following discrete-time linear-Gaussian dynamical system,

$$\mathbf{X}_{j+1} = A_d \mathbf{X}_j + \mathbf{W}_j \tag{2.30}$$

where $\mathbf{X}_j \equiv \mathbf{X}(j \cdot \Delta t)$ for $j \geq 0$, $\mathbf{W}_j \sim N(0, Q_d)$, and,

$$A_d = \exp[\Delta t\, A] \tag{2.31}$$

$$Q_d = \int_0^{\Delta t} \exp[\tau\, A] Q \exp[\tau\, A^T] d\tau, \tag{2.32}$$

where $\exp[\cdot]$ is the matrix exponential function.

The expression for A_d can be derived by multiplying Equation (2.29) through by $\exp[-tA]$ and then solving for $\mathbf{X}(t)$. The expression for Q_d can be derived by evaluating $\mathrm{E}[\mathbf{X}(t)\mathbf{X}(t+\Delta t)^T]$. Furthermore, Q_d can be evaluating by first computing,

$$F = \exp\left(\Delta t \begin{bmatrix} -A & Q \\ 0 & A^T \end{bmatrix} \right) = \begin{bmatrix} F_{11} & F_{12} \\ 0 & F_{22} \end{bmatrix} \tag{2.33}$$

where F_{ij} is a matrix with the same dimensions as A. Then [Van Loan, 1978],

$$Q_d = F_{22}^T F_{12}. \tag{2.34}$$

Finding analytical discretizations or numerical schemes for approximately discretizing nonlinear SDEs is a challenging problem. Analytical solutions tend to be specific to certain classes of model, and do not apply to the models we are interested in, as far as we are aware, see for example Section 3.2 of [Andrieu et al., 2010]. Further exposition would also require the introduction of concepts from stochastic calculus. The interested reader is referred to Chapter 3 of [Maskell, 2004].

Numerical schemes for approximating nonlinear SDEs also tend to be specific to certain types of SDE. One of the most commonly used schemes is the Euler-Maruyama method. This can be

applied to nonlinear SDEs with additive noise. For simplicity we consider SDEs with additive uncorrelated white noise,

$$d\mathbf{X}(t) = \mathbf{\Phi}[\mathbf{X}(t)]dt + d\mathbf{W}(t), \tag{2.35}$$

where $\mathbf{W}(t)$ is again a d_x dimensional Wiener process with uncorrelated components. Such an SDE can be discretized using the Euler-Maruyama method resulting in an approximate discrete-time process. In order to approximate $\mathbf{X}((j+1)\cdot\Delta t)$ given an approximation to $\mathbf{X}(j\cdot\Delta t)$ we need to perform K iterations of the Euler-Maruyama scheme,

$$\mathbf{X}_{k+1} = \mathbf{X}_k + h\mathbf{\Phi}[\mathbf{X}_k] + \sqrt{h}\mathbf{W}_k \tag{2.36}$$

for $k = 1, \ldots, K$, $h = \Delta t/K$, and $\mathbf{X}_1 \approx \mathbf{X}(j\cdot\Delta t)$, i.e., the initial state is the approximation to $\mathbf{X}(j\cdot\Delta t)$.

The accuracy of the Euler-Maruyama scheme depends on the numerical time-step in the solver, h. Alternatively it is also possible to reduce the discretization error by using a different SDE numerical solver, as for example in [Murray, 2013].

If we choose the time-step in the Euler-Maruyama solver to be $h = \Delta t$ this may result in a poor approximation, or even qualitatively different dynamics in terms of stability. Instead, when Δt is relatively large we may choose $K \gg 1$ to ensure that h is sufficiently small.

Note that when $K > 1$ the Euler-Maruyama scheme results in a discrete-time dynamical system where the approximation to $\mathbf{X}((j+1)\cdot\Delta t)$ is not an additive function of the approximation to $\mathbf{X}(j\cdot\Delta t)$ and $\mathbf{W}_1, \ldots, \mathbf{W}_K$. This means that Kalman filters such as the EKF and the UKF cannot be applied to this discretization for $K > 1$.

2.6.2 *Kalman filter marginal likelihood for linear systems*

The basic Kalman filter is a method for linear state space models that evolves a probability distribution over system states forward in time and updates this distribution using Bayes' rule at observation time-points. Assuming a Gaussian prior, the distribution over states given ob-

servations is Gaussian. The Kalman filter specifies the mean and covariance of this distribution [Bishop, 2006]. The Kalman filter is commonly used to estimate the current state of the system (often referred to as the filtering distribution) and to predict future states of the system. It can also be used to calculate the marginal likelihood of the observations analytically by integrating out the state space variables.

We are interested in problems where we have $10 - 50$ system parameters and we observe the system at $\sim 10^4$ time-points. Integrating out the state variables can make exploration of the parameter space significantly more tractable. For example an MCMC algorithm that explores the marginal parameter space will typically converge much faster than an MCMC algorithm that explores the joint distribution of system parameters and the latent state space variables.

The marginal likelihood of $\mathbf{y}_{1:n}$, conditional on θ is,

$$p_\theta(\mathbf{y}_{1:n}) = p_\theta(\mathbf{y}_1) \prod_{j=2}^{n} p_\theta(\mathbf{y}_j|\mathbf{y}_{1:j-1}), \tag{2.37}$$

where

$$p_\theta(\mathbf{y}_j|\mathbf{y}_{1:j-1}) = \int g_\theta(\mathbf{y}_j|\mathbf{x}_j) f_\theta(\mathbf{x}_j|\mathbf{x}_{j-1}) p_\theta(\mathbf{x}_{1:j-1}|\mathbf{y}_{1:j-1})\, dx_{1:j}, \tag{2.38}$$

and g_θ is the conditional probability distribution of observations given state variables, f_θ is the transition density for the state variables, and the p_θ are also probability densities.

We now write down the equations which can be used to evaluate this marginal likelihood analytically. Let,

$$\boldsymbol{\mu}_0 = \mathrm{E}[\mathbf{X}_1] \tag{2.39}$$

$$V_0 = \mathrm{Cov}[\mathbf{X}_1] \tag{2.40}$$

Then [Bishop, 2006, p. 639],

$$P_0 = AV_0A^T + Q \tag{2.41}$$

$$K = P_0H^T(HP_0H^T + \Sigma)^{-1} \tag{2.42}$$

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}_j] = A\boldsymbol{\mu}_0 + K(\mathbf{y} - HA\boldsymbol{\mu}_0) \tag{2.43}$$

$$V = \mathrm{Cov}[\mathbf{X}_j] = (I - KH)P_0 \tag{2.44}$$

$$c_n = p_\theta(\mathbf{y}_j|\mathbf{y}_{1:j-1}) = \mathrm{N}(\mathbf{y}; HA\boldsymbol{\mu}_0, HP_0H^T + \Sigma). \tag{2.45}$$

where N is the probability density function of a multivariate normal distribution.

If we want to evolve the probability distribution forward one time-step and there are no observations to account for, the equations simplify to,

$$P_0 = AV_0A^T + Q \tag{2.46}$$

$$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}_j] = A\boldsymbol{\mu}_0 \tag{2.47}$$

$$V = \mathrm{Cov}[\mathbf{X}_j] = P_0 \tag{2.48}$$

The computational complexity of calculating each term in the marginal likelihood as written in Equation (2.37) is $O({d_x}^3)$ because of the matrix multiplication in Equation (2.41). So, if there are n observations the marginal likelihood calculation has a total computational cost of $O(n{d_x}^3)$.

2.6.3 *Particle filters for nonlinear systems*

The particle filter is a Monte Carlo method for approximating the probability distribution over system states in nonlinear SSMs. Similarly to the Kalman filter it rests on the use of Bayes' rule to update beliefs about the system state in light of observations.

Given an initial set of particles $\{\mathbf{x}_1^{(i)} : i = 1, \ldots, N\}$ and weights $\{w_1^{(i)} : i = 1, \ldots, N\}$, and a proposal distribution, $q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$, each iteration in time of the particle filter can be described as follows [Van Der Merwe et al., 2001]:

i. Sequential importance sampling step:

- for $i = 1, \ldots, N$, sample $\tilde{\mathbf{x}}_t^{(i)} \sim q(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})$,
- for $i = 1, \ldots, N$, evaluate the importance weights up to a normalizing constant

$$w_t^{(i)} = \frac{p(\mathbf{y}_t|\mathbf{y}_{1:t-1}, \mathbf{x}_{1:t})p(\mathbf{x}_t|\mathbf{x}_{t-1})}{q(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{x}_{1:t-1})}, \tag{2.49}$$

- for $i, \ldots, N$, normalize the weights: $W_t^{(i)} = w_t^{(i)} \left[\sum_{j=1}^N w_t^{(i)}\right]^{-1}$.

ii. Selection step:

- multiply / suppress samples $\{\tilde{\mathbf{x}}_t^{(i)} : i = 1, \ldots, N\}$ with high/low importance weights $W_t^{(i)}$ respectively, to obtain N random samples $\{\bar{\mathbf{x}}_t : i = 1, \ldots, N\}$ approximately distributed according to $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

The posterior density of the state variables can then be approximated as,

$$\hat{p}_\theta(d\mathbf{x}_{1:n}|\mathbf{y}_{1:n}) = \sum_{i=1}^N W_n^{(i)} \delta_{\mathbf{X}_{1:n}^{(i)}}(dx_{1:n}), \tag{2.50}$$

where $\delta_{\mathbf{X}_{1:n}^{(i)}}(dx_{1:n})$ is a multivariate Dirac delta function, re-centred at the point $\mathbf{x}_{1:n}^{(i)}$.

And the sequence of terms required for estimating the marginal likelihood is,

$$\hat{p}_\theta(\mathbf{y}_n|\mathbf{y}_{1:n-1}) = \frac{1}{N}\sum_{i=1}^N w_n^{(i)}. \tag{2.51}$$

If we use N particles to estimate the integrals in Equation (2.38), the overall cost of the estimate is $O(Nnd)$, where d is the dimension of the state-space. Depending on the problem and the variant of particle filter used, N may be somewhere in the range 100-10,000. Outside of special cases where we may simulate exactly from the SDE dynamics, this approach yields a biased approximation to the likelihood due to discretization error. If we wish for a better approximation to the SDE, we may sample paths on a finer grid than the observations, which increases the computational cost of the particle filter.

Some common choices of proposal distribution are,

- Bootstrap. Set $q(\mathbf{x}_t|\mathbf{y}_{1:t}, \mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. This choice of proposal, and the function $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ are sometimes referred to as the transition density prior.

- Unscented Kalman Filter (UKF). Set $q(\mathbf{x}_t^{(i)}|\mathbf{y}_{1:t}, \mathbf{x}_{1:t-1}^{(i)}) = \mathrm{N}(\boldsymbol{\mu}_t^{(i)}, P_t^{(i)})$ where N denotes a multivariate Normal distribution and $\boldsymbol{\mu}_t^{(i)}, P_t^{(i)}$ are determined by applying a UKF to each particle.

If observations are highly informative with respect to the state of the system, the UKF proposal will perform better than the basic bootstrap proposal because the UKF takes account of $\mathbf{y}_t$ whereas the bootstrap proposal is independent of $\mathbf{y}_t$. However the UKF proposal can only be constructed when the nonlinear state-space model is additive (i.e. can be written in the form of Equation (2.26)).

For the bootstrap particle filter we have the following expression for the weights,

$$w_n^{(i)} = \prod_{j=0}^{n} p_\theta(\mathbf{y}_j|\mathbf{x}_j^{(i)}) \tag{2.52}$$

The expected value of w_n is an estimate for the marginal likelihood,

$$\mathrm{E}[w_n] = \int \left(\prod_{j=1}^{n} p_\theta(\mathbf{y}_j|\mathbf{x}_j)\right) p_\theta(\mathbf{x}_1) \left(\prod_{j=2}^{n} p_\theta(\mathbf{x}_n|\mathbf{x}_{n-1})\right) dx_{0:n} \tag{2.53}$$

$$= \int p_\theta(\mathbf{y}_{0:n}|\mathbf{x}_{0:n}) p_\theta(\mathbf{x}_{0:n}) dx_{0:n} \tag{2.54}$$

$$= p_\theta(\mathbf{y}_{0:n}) \tag{2.55}$$

Therefore $\frac{1}{N}\sum_{i=1}^{N} w_n^{(i)}$ will be a good estimate for $p_\theta(\mathbf{y}_{0:n})$ for N sufficiently large.

If we just use importance sampling (i.e. SMC without resampling) with the prior dynamics as the proposal density, the relative variance in the marginal likelihood estimate ($\mathrm{Var}[\hat{p}_\theta(\mathbf{y}_{0:n})/p_\theta(\mathbf{y}_{0:n})]$) increases exponentially with n. In contrast, under relatively mild conditions, the relative variance increases linearly with n in standard SMC algorithms,

$$\mathrm{Var}[\hat{p}_\theta(\mathbf{y}_{0:n})/p_\theta(\mathbf{y}_{0:n})] < \frac{D_\theta n}{N}. \tag{2.56}$$

A tutorial that includes further discussion of these theoretical results can be found in [Doucet and Johansen, 2009].

As well as n, another key problem parameter in many applications is d_y, the dimension of $\mathbf{y}_n$. The variance of marginal likelihood estimates (and other estimators of interest) obtained using SMC typically grows exponentially with d_y. Some progress has been made towards solving this problem, [Beskos et al., 2014, Rebeschini and Van Handel, 2015], but it remains a large obstacle in using SMC for estimation for discretizations of stochastic PDEs and for many SDE systems.

2.6.4 *Particle MCMC for static parameter estimation*

The static parameters of state-space models can be sampled using particle MCMC algorithms. For example, the Particle Marginal Metropolis Hastings (PMMH) algorithm is a pseudo-marginal algorithm where a particle filter estimate of the marginal likelihood, based on Equation (2.51), is used [Andrieu et al., 2010]. Estimating the marginal likelihood with a small number of particles will occasionally result in substantial over-estimation of the marginal likelihood. Whenever this happens, the MCMC chain will tend to get stuck because the over-estimated marginal likelihood will bias the acceptance probability towards zero. It has been found that the optimal number of particles is such that the variance in the marginal likelihood is $O(1)$, [Doucet et al., 2015, Nemeth et al., 2016a].

The effect of the dimension of the observation space, d_y, the number of observations n, and the informativeness of the observations on the variance of marginal likelihood estimates means that as these factors increase, the number of particles required to ensure that the variance is $O(1)$ increases rapidly, and so the cost of particle MCMC can quickly become prohibitive. The computational cost can sometimes be made tractable by using fast implementations of SDE solvers and parallelizing across particles. These features are implemented in the Libbi software [Murray, 2013], so can be used without placing too much burden on the user, at least in principle.

As well as estimating the marginal likelihood, the particle filter can also be used to estimate the gradient and Hessian of the marginal likelihood [Poyiadjis et al., 2011, Nemeth et al., 2016a].

This has lead to variants of particle MCMC that use Langevin dynamics to explore the parameter space [Dahlin et al., 2015, Nemeth et al., 2016b].

Despite much progress in this area, the current limits in problem complexity are (roughly) up to 10 state variables, up to 5 unknown static parameters, and $O(10^2)$ observations, although this is somewhat dependent on the computing platform and the software implementation. For a more general overview of parameter estimation in state-space models using particle filters, see [Kantas et al., 2015]

2.7 CONTEXTUAL SUMMARY

In this chapter we have seen a variety of methods that are used to approximate high-dimensional integrals in Bayesian inference. In Section 2.2 we covered approximate inference methods that are deterministic and relatively fast. The accuracy of these methods depends heavily on the problem. If the posterior distribution is close to Gaussian, for example, then approximate methods will be relatively accurate. In Section 2.3 we covered MCMC methods. Estimators produced using MCMC are random variables. Under certain assumptions, these estimates are unbiased and converge to the true underlying value with the number of MCMC iterations. The computational efficiency of MCMC depends heavily on the problem and on the variant of MCMC that is used. Recently developed samplers based on discretizing Langevin and Hamiltonian dynamics are well suited to problems where the posterior is non-Gaussian and also scale well to high-dimensional parameters spaces. The main drawback of these methods is the requirement to evaluate derivatives which may be computationally costly and/or time-consuming for users. If the posterior is low dimensional and dependencies between parameters in the posterior distribution are relatively weak then Gibbs sampling or Metropolis-within-Gibbs may be more computationally efficient and easier to implement than methods that require local derivative information. In Section 2.4 we covered SMC, a very general class of methods which can address some of the limitations of MCMC, such as being able to explore multimodal distributions more efficiently and produce marginal likelihood estimates. In Section 2.5 where SMC estimates are

used in place of intractable likelihoods. And in Section 2.6 we considered methods that are specific to state-space models.

At the beginning of this chapter we said that we were interested in models where parameters can be partitioned into θ and x where the dimension of x is much larger than θ, and in problems where the primary interest is in obtaining samples from the posterior distribution of θ given some observations y. In particular this type of problem arises in parameter estimation problems for dynamical systems from time-series data, where θ represent parameters that determine the dynamics of the model, x is a series of multivariate state variables. Pseudo-marginal algorithms, described in Section 2.5, are a recent advance in Computational Statistics that offers one possible solution to this problem. Particle MCMC (discussed in Section 2.6.4) is an example of a pseudo-marginal algorithm designed for parameter estimation in dynamical systems and is one possible solution to the problem we are interested in. The main attractions of particle MCMC are that it comes with theoretical guarantees that estimates are unbiased. By using a particle filter to estimate the marginal likelihood of parameters given a time-series it is more computationally efficient than more basic MCMC approaches sample the joint (θ, x) space. However, as discussed above, the computational cost of particle MCMC does not scale well with the number of unknown parameters, the number of observation variables, or the length of the time-series. In Section 2.6.2 we discussed linear-Gaussian state-space models, where the marginal likelihood can be calculated analytically. In this case it is possible to implement MCMC with exact marginal likelihoods instead of particle filter estimates, and this is also more computationally efficient than particle MCMC.

In Chapters 4 to 7 we focus on the models where the dynamics are approximately linear. In Chapter 4 we introduce another class of methods based on spectral analysis that can be used to estimate marginal likelihoods and can be more computationally efficient than Kalman filters. Compared to the EKF, which linearizes the dynamics at each iteration of the filter, spectral methods require the model to be linearized around a stable equilibrium point of the model (as

described in Section 4.1.3). Furthermore spectral methods introduce an asymptotic approximation in the length of the time-series. We will see in Chapter 4 that there are cases where the error introduced by these approximations is negligible. However Kalman filters and particle filters have an important role to play in quantifying this error. In Chapter 5 we introduce a novel way of parameterizing models with a stable equilibrium. The reparameterization can be applied in the context of any MCMC algorithm, and does not introduce any further approximations. We will see that effective reparameterization leads to more computationally efficient MCMC samplers. Chapter 6 gives an overview of my C++ implementation of the methods developed in Chapters 4 and 5. And Chapter 7 applies these methods to an EEG data analysis problem in Neuroscience.

3

NEURAL POPULATION MODELS

Much effort has been devoted to understanding the physical mechanisms underlying neural activity. This has led to mathematical models at a number of different description levels ranging single neurons up to macroscopic brain regions. Our primary interest is in mesoscopic models, which integrate properties of microscopic neurophysiology with the mechanisms that lead to macroscopic recording of brain activity, such as EEG.

In principle mesoscopic models can be used to make predictions, for example, what effect do anaesthetic drugs have on neural activity? Mesoscopic models can also be used to formulate inverse problems, i.e., given an EEG dataset, what is the underlying state of neural activity? In practice these types of question are difficult to answer because the behaviour of neural models is sensitive to parameter values, and such parameters are only known within relatively broad ranges from independent experiments. The behaviour of mesoscopic models is also sensitive to the modelling assumptions which determine the form of the mesoscopic model. In mesoscopic modelling and Computational Neuroscience more generally there is often a range of different assumptions that could plausibly be made.

The effect of anaesthetic agents on neural activity at the microscopic level is relatively well understood. Furthermore, the neuroimaging correlates of anaesthesia are well established. This creates an opportunity for constraining the parameter values of mesoscopic models and for

testing the effect of different modelling assumptions. Solving this relatively tractable problem is still challenging and forms much of the motivation behind subsequent chapters.

This chapter is built around introducing the Liley *et al* Neural Population Model (NPM), which we introduce in the context of excitatory-inhibitory networks (Section 3.1.4). Prior to that we touch on the physiology of individual neurons (Section 3.1.1) and of synapses (Section 3.1.2). This is followed by a brief discussion of the measurement physics that connect neural activity to macroscopic signals, such as the EEG, via electric field potentials (Section 3.2). And then a review of the physiology and neuroimaging of anaesthesia with an emphasis on modelling the effect of the anaesthetic isoflurane (Section 3.3).

Note that this chapter does not constitute a comprehensive review of NPMs. Different models that have a similar underlying structure, have been developed to model the effect of propofol on the EEG [Hutt and Longtin, 2010], and the changes in brain physiology that occur during sleep [Robinson et al., 2001], to select two further examples from the wider literature.

3.1 NEUROPHYSIOLOGY

3.1.1 *Electrophysiology*

The membrane potential dynamics of an individual neuron can be modelled in terms of the current flows through the cell membrane. The magnitudes of ionic currents depend on the proportion of ion channels on the cell membrane that are open. The ion channel dynamics depend on the membrane potential, leading to a system of coupled nonlinear differential equations.

There are many different variations of electrophysiology models. Perhaps the most celebrated model is that of Hodgkin and Huxley, which was developed to describe action potentials in the giant axon of the squid [Hodgkin and Huxley, 1952], and led to a Nobel Prize for Hodgkin and Huxley in 1963.

Symbol	Description	Units	L value	Na value	K value
E	reversal potential	mV	-80	60	-90
g	maximal conductance	$m\Omega^{-1}$ / cm^2	8	20	10
μ	activation threshold	mV	-	-20	-45
σ	inverse activation slope	mV	-	15	5
τ	activation rate	ms	-	-	1

Table 3.1.: Parameters of $I_{Na} + I_K$ model.

The key properties of the Hodgkin-Huxley equations and related models can be illustrated using a simple $I_{Na} + I_K$ model,

$$\begin{aligned} \frac{dV}{dt} &= I - I_L(V) - I_{Na}(V) - I_K(n, V) \\ \frac{dn}{dt} &= (n_\infty(V) - n)/\tau \end{aligned} \tag{3.1}$$

where

$$\begin{aligned} I_L(V) &= g_L(V - E_L) \\ I_{Na}(V) &= g_{Na} m_\infty(V)(V - E_{Na}) \\ I_K(n, V) &= g_K n(V - E_K) \end{aligned} \tag{3.2}$$

and

$$m_\infty(V) = \frac{1}{1 + \exp[-(V - \mu_{Na})/\sigma_{Na}]} \tag{3.3}$$

$$n_\infty(V) = \frac{1}{1 + \exp[-(V - \mu_K)/\sigma_K]}. \tag{3.4}$$

The variable V represents the electrical potential across the cell membrane, and the variables m, n represent the proportions of Sodium and Potassium ion channels that are open. These proportions determine the conductances for Sodium and Potassium currents across the cell membrane. In this model the Potassium current, $I_K(V)$, is a slow current because of the time it takes for n to get close to its equilibrium value $n_\infty(V)$. In contrast the Sodium current, $I_{Na}(V)$, is fast because m reaches its equilibrium, $m_\infty(V)$, instantaneously. The leakage current, $I_L(V)$,

does not depend on ion channel dynamics. The parameters of the $I_{Na} + I_K$ model are described in Table 3.1 and values are given for the low threshold variant of the model. (In the high threshold variant, $\mu_K = -25$ and the model has qualitatively different behaviour.) A longer exposition of the $I_{Na} + I_K$ model can be found in Chapter 4 of [Izhikevich, 2007].

When the injected current, I, is zero the low threshold $I_{Na} + I_K$ model has a stable equilibrium (and no other attractor states). When the system is at its stable equilibrium we say that the neuron is at rest. Above a certain threshold of injected current, e.g. $I = 40$, the system has a stable limit cycle. (In contrast, the stable equilibrium and the stable limit cycle can co-exist in the high threshold variant.) In Neuroscience nonlinear oscillations are referred to as action potentials. And when the system produces a sequence of action potentials we say that the neuron is firing.

Given the parameter values in Table 3.1, $I_{Na}(V)$ is negative and $I_K(V)$ is positive for $V \in [-90, 60]$. The sign of dV/dt depends on n (as well as V). During the upward phase of an oscillation n is relatively low, so the inward I_{Na} current outweighs the I_K current, and $dV/dt > 0$. The increase in V causes n to increase (on a slower timescale) leading to the downward phase of the oscillation where the outward I_K current is larger.

There is not an intuitive explanation why some parameters values lead to a stable equilibrium (i.e. a resting neuron) and some parameter values lead to a stable limit cycle (i.e. a firing neuron). However, for a given set of parameters it is possible to evaluate the stability of the fixed point (V^*, n^*) by linearizing the model around the fixed point and evaluating the eigenvalues of the linearized system. In general parameter sets with higher V^* are more likely to be unstable and are associated with limit cycles. This observation has lead to the development of Integrate and Fire models where all the ionic currents are linear and action potentials are modelled by simply resetting the value of the membrane potential when it reaches some threshold, see Chapter 8 of [Izhikevich, 2007] for more details.

The FitzHugh-Nagumo (FHN) model is a simpler model than the $I_{Na} + I_K$ model in that it only has four parameters. The FitzHugh Nagumo equations are only loosely connected to electrophysiology, but the model has proved to be popular among computational neuroscientists due to the tractability of mathematical analysis.

The FitzHugh-Nagumo equations are,

$$\frac{dV}{dt} = I - I_{in}(V) - I_{out}(w) \tag{3.5}$$

$$\frac{dw}{dt} = bV - cw \tag{3.6}$$

where

$$I_{in}(V) = -V(a - V)(V - 1) \tag{3.7}$$

$$I_{out}(w) = w \tag{3.8}$$

The FitzHugh Nagumo model differs from the $I_{Na} + I_K$ model in that the fast inward Na current, $I_{Na}(V)$ has been replaced by a cubic polynomial, $I_{in}(V)$, and the slow outward K current, $I_K(n, V)$ has been replaced by a linear outward current, $I_{out}(w)$. Although not immediately obvious from the model equations, the V nullcline (the curve along which $dV/dt = 0$) for the $I_{Na} + I_K$ model is a cubic polynomial. It is straight-forward to show that the V nullcline in the FitzHugh Nagumo equations is a cubic polynomial, and it is this similarity between the models that leads to them having qualitatively similar dynamics. Similarly to the low threshold $I_{Na} + I_K$ model, the FitzHugh Nagumo equations can have a single stable equilibrium (and no other attractors). And above a certain threshold of I, the system has a stable limit cycle. More details can be found in Chapter 4 of [Izhikevich, 2007].

3.1.2 *Synaptic physiology*

A synapse is a connection between two neurons. When a pre-synaptic neuron fires an action potential, neurotransmitters are released into the extracellular space. These neurotransmitters

bind to receptors on the post-synaptic neuron. This binding triggers an electrophysiological response in the post-synaptic neuron.

Synapses can be inhibitory or excitatory. At most inhibitory synapses the $GABA_A$ neurotransmitter is released and binds to $GABA_A$ receptors on the post-synaptic neuron. This leads to an outward Cl^- current. At most excitatory synapses the glutamate neurotransmitter is released and binds to post-synaptic receptors leading to an inward ionic current. The primary receptors at excitatory synapses in cortex are AMPA and kainate receptors. See Chapter 3 of [Kalat, 2015] for more details.

In contrast to the previous section (and the wider literature on single cell electrophysiology) where a sequence of action potentials was described in terms of nonlinear dynamics, we now reduce the description down to a collection of time points at which action potentials occur. This sequence of times is referred to as a spike train. Later we will describe pre-synaptic input through a firing rate. Our primary interest now is in deriving a model for the post-synaptic response, $I(t)$, to pre-synaptic input, $f(t)$.

The response of a post-synaptic neuron to a single pre-synaptic spike at time $t = 0$ can be described by a post-synaptic response function. One commonly used response function is the following, sometimes referred to as an α-function,

$$R_0(t) = \gamma^2 t \exp[-\gamma t]\Theta(t), \tag{3.9}$$

where $\Theta(t)$ is the Heaviside function, defined as,

$$\Theta(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ \\ 0 & \text{if } t < 0. \end{cases}$$

The parameter γ is called the shape parameter because it changes the shape of $R_0(t)$ without changing the total area under the curve. In Section 3.1.4 we will add another parameter Γ, to control the area under the curve.

Using Fourier analysis, it can be shown that $R_0(t-t_k)$ is the solution to the following differential equation,

$$\left(\frac{d}{dt}+\gamma\right)^2 G(t;t_k) = \gamma^2\delta(t-t_k), \tag{3.10}$$

where $\delta(t)$ is the Dirac δ-function, which is the distributional derivative of the Heaviside function, and has the property,

$$\int_{t_1}^{t_2}\delta(t)dt = \begin{cases} 1 & \text{if } 0 \in (t_1,t_2) \\ 0 & \text{if } 0 \notin (t_1,t_2). \end{cases}$$

By changing $\delta(t)$ in equation (3.10) to the more general $f(t)$ we obtain a differential equation that describes the post-synaptic response to general pre-synaptic input, as opposed to a single spike,

$$\left(\frac{d}{dt}+\gamma\right)^2 I(t) = \gamma^2 f(t). \tag{3.11}$$

The function, $G(t;t_k) = R_0(t-t_k)$ is referred to as the Green's function of the differential equation (3.11). We can write the solution to Equation (3.11) as the homogeneous solution plus the convolution of, $\gamma^2 f(t)$ with the Green's function,

$$I(t) = I(0)\exp[-\gamma t] + \gamma^2\int_0^t R_0(t-s)f(s)ds. \tag{3.12}$$

In what follows we neglect the homogeneous term $I(0)\exp[-\gamma t]$ as we are typically interested in the behaviour of the system when it is in its stationary phase.

If we knew that spikes took place at $t_1,\ldots,t_K$, then $f(t)$ would just be a sum of δ-functions and, by equation (3.12), the solution to equation (3.11) would be $I(t) = \sum_{k=1}^K R_0(t-t_k)$.

A mean-field model is obtained by considering the activity of a large population of neurons, and modelling only the mean activity of that population, rather than the activity of each individual neuron in the population.

In mean-field models, we assume there is a large number of pre-synaptic neurons, N, each of which has an average firing rate, $S(t)$. The firing rate tells us how many spikes we can expect to see within a given time window from a single pre-synaptic neuron.

In order to justify the deterministic mean-field model we now analyse $I(t)$ under two different forms of input. First we model the input as a random spike train, i.e., a sum of δ functions centred on a random sequence of time points controlled by the firing rate $S(t)$. Then we model the input as the deterministic function $S(t)$.

Let M_i be a random variable representing the number of spikes in the interval $[t, t+\delta t]$ from the ith pre-synaptic neuron. Then,

$$\mathrm{E}[M_i] = \int_t^{t+\delta t} S(t)dt \approx \delta t S(t) \text{ for } \delta t \text{ small.}$$

If neurons fire independently of each other, the expected total number of spikes coming from the N input neurons is,

$$\mathrm{E}\left[\sum_{i=1}^{N} M_i\right] \approx N\delta t S(t).$$

We can decompose the convolution into a sum over time bins,

$$I(t) = \gamma^2 \sum_{j=1}^{J} \int_{s_j}^{s_{j+1}} R_0(t-s)f(s)ds,$$

where $s_1 = 0$ and $s_J = 1$, and $s_{j+1} - s_j = \delta s$. Each term in the sum quantifies the influence of $f(s)$ in the time interval $[s_j, s_{j+1}]$ on the synaptic response at time t.

Now consider the expected value of each term in the sum receiving input from N random spike trains with firing rate, $S(t)$, and spike times $\{s_k^{(i)}\}$, where $s_k^{(i)}$ is the time of the kth spike from the ith neuron in the time window $[s_j, s_{j+1}]$,

$$\begin{aligned}\mathrm{E}\left[\gamma^2 \int_{s_j}^{s_{j+1}} R_0(t-s) f(s) ds\right] &\approx \gamma^2 R_0(t-s) \mathrm{E}\left[\int_{s_j}^{s_{j+1}} \sum_{i=1}^{N} \sum_{k=1}^{M_i} \delta(s - s_k^{(i)}) ds\right] \\ &\approx \gamma^2 R_0(t-s_j) S(s_j) N \delta s.\end{aligned}$$

The smaller $s_{j+1} - s_j$ is, the better the approximation will be.

Under the deterministic firing rate model we have,

$$\left(\frac{d}{dt} + \gamma\right)^2 I(t) = \gamma^2 N S(t) \tag{3.13}$$

and

$$I(t) = \gamma^2 \sum_{j=1}^{J} \int_{s_j}^{s_{j+1}} R_0(t-s) N S(s) ds,$$

with

$$\gamma^2 \int_{s_j}^{s_{j+1}} R_0(t-s) N S(s) ds \approx \gamma^2 R_0(t-s_j) S(s_j) N \delta s.$$

As discussed earlier, the expected number of spikes in a small time interval around s_j is $S(s_j)\delta s$. The derivation above demonstrates that the expected value of $I(t)$ for the model with random spike train input is equal to $I(t)$ for the model with deterministic firing rate input.

In the mean-field approximation the higher-order (≥ 2) moments of $I(t)$ are ignored. Pre-synaptic activity is described through the firing rate rather than the spike train, and equation (3.13) is used to model the post-synaptic response to pre-synaptic activity. Ignoring higher-order moments means, for example, that the variance in the activity of one population cannot influence the mean activity of another population. The main alternatives that avoid the mean-field approximation are the Fokker-Planck equation which models the evolution of the distribution

of population activity through a PDE, and spiking neuronal networks where the activity of each neuron is modelled. Simulation and data assimilation are more intensive for these alternatives. As a result the mean-field approach has proved the most popular in the context of relating recordings of macroscopic neural activity (e.g. through the EEG) to microscopic and mesoscopic biophysical parameters. Fuller discussion of the assumptions underpinning the mean-field approximation and mathematical methods for studying population activity can be found in [Deco et al., 2008, Bressloff, 2011].

The equations we have presented so far are useful for describing small regions of cortex where we can assume that connectivity is global and homogeneous, and communication is effectively instantaneous. The small sub-regions of the cortex where these assumptions hold are referred to as macro-columns. Between macro-columns, there is long-range excitatory connectivity. The strength of this connectivity depends on the distance between macro-columns. This is parametrized by a cortico-cortical decay scale, Λ. Communication between macro-columns is propagated with a velocity, v, so that the time taken for activity in one macro-column to influence activity in another macro-column depends on the distance between the macro-columns.

Above we considered the post-synaptic response to a single pre-synaptic spike. This response was a function of time. Now we consider the spatial propagation of electrical activity in response to a pre-synaptic spike. We model a spike as an event taking place at a discrete point in space at a particular time point. The spatial response function that describes travelling waves is,

$$R_s(r,t) = \frac{\Lambda}{2\pi} \exp\left(-v\Lambda t\right)\ \delta\left(t - \frac{r}{v}\right), \tag{3.14}$$

where r represents distance from the spike, and t represents time after the spike.

This response function says that $R_s(r,t) = 0$, except when $r = vt$. The exponential decay term ensures that the amplitude of the response gets smaller (as well as later) as we move further away from the source. Requiring $R_s(r,t) = 0$, when $r \neq vt$ effectively says that conduction times are infinitely precise. This is an approximation that can be relaxed using the methods

described in [Bojak and Liley, 2010]. However spatiotemporal response functions that more accurately reproduce patterns of spatial propagation seen in experiments lead to PDEs that are very challenging to solve numerically.

The spatiotemporal response of any system, $\Phi(\mathbf{x}, t)$ can be described as the convolution of a Green's function, $G(\mathbf{x}, t; \mathbf{y}, s)$, with an input function $f(\mathbf{x}, t)$,

$$\Phi(\mathbf{x}, t) = \int_0^T \int_C G(\mathbf{x}, t; \mathbf{y}, s) f(\mathbf{y}, s) \; d\mathbf{y} \; ds. \tag{3.15}$$

In our model, single spikes are represented as δ-functions, and the Green's function is $G(\mathbf{x}, t; \mathbf{y}, s) = R_s(|\mathbf{x} - \mathbf{y}|, t - s)$. If we knew the locations and timings of a spatiotemporal spike train, $\{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_K, t_K)\}$, the solution to equation (3.15) would be,

$$\Phi(\mathbf{x}, t) = \sum_{k=1}^{K} R_s(|\mathbf{x} - \mathbf{x}_k|, t - t_k).$$

In the non-spatial setting we averaged over time. In the spatial setting we average over time and space. Consider the expected value of $\Phi(\mathbf{x}, t)$ for input in a small neighbourhood around $(\mathbf{y}_0, s_0)$,

$$\begin{aligned}
&\mathrm{E}\left[\int_{s_0}^{s_0+\delta s} \int_{\Delta C} R_s(|\mathbf{x} - \mathbf{y}|, t - s) f(\mathbf{y}, s) \; d\mathbf{y} \; ds\right] \\
&\approx R_s(|\mathbf{x} - \mathbf{y}_0|, t - s_0) \mathrm{E}\left[\int_t^{t+\delta t} \int_{\Delta C} \sum_{i=1}^{N} \sum_{k=1}^{M_i} \delta(\mathbf{y} - \mathbf{y}_k^{(i)}, s - s_k^{(i)}) \; d\mathbf{y} \; ds\right] \\
&\approx R_s(|\mathbf{x} - \mathbf{y}_0|, t - s_0) \, S(\mathbf{y}_0, s_0) \, N \delta s (\delta y)^n,
\end{aligned}$$

where ΔC is a hypercube of length δy and dimension n centred at $\mathbf{y}_0$. The scalar N is now the number of long-range connections between macro-columns, and $S(\mathbf{x}, t)$ is the average firing rate of a single neuron at location $\mathbf{x}$ and time t.

Under the deterministic firing rate model we have,

$$\Phi(\mathbf{x}, t) = \int_0^T \int_C R_s(|\mathbf{x} - \mathbf{y}|, t - s) \; NS[h(\mathbf{y}, s)] \; d\mathbf{y} \; ds. \tag{3.16}$$

Analogously to the non-spatial case it should now be clear that the expectation of the post-synaptic response in a random spike train model is same as the post-synaptic response in a deterministic firing rate model.

Using Fourier analysis it can be shown that if $R_s(r,t)$ is the response function, and $S[h(\mathbf{x},t)]$ is the pre-synaptic firing rate, the solution to the integral equation (3.16) can be approximated by the solution to the following PDE,

$$\left[\left(\frac{\partial}{\partial t}+v\Lambda\right)^2-\frac{3}{2}v^2\nabla^2\right]\Phi(\mathbf{x},t)=v^2\Lambda^2NS[h(\mathbf{x},t)]. \tag{3.17}$$

where ∇^2 is the spatial Laplacian, and the cortex is typically modelled as a 2-D sheet, in which case $\mathbf{x}=(x,y)$. The idea of using Green's functions to convert integral equations into PDEs in Computational Neuroscience was introduced in [Jirsa and Haken, 1996], although there are earlier instances, [Griffith, 1963].

3.1.3 *System dynamics*

For a single synaptic current, the mean membrane potential dynamics are modelled by the following equation,

$$\tau\frac{dh(t)}{dt}=-(h(t)-h^r)+\psi[h(t)]I(t). \tag{3.18}$$

The function $\psi[h(t)]$ is defined in equation (3.19) and the parameter h^r is referred to as the resting potential. In the absence of any synaptic currents, the mean membrane potential dynamics will just be an exponential decay towards h^r.

In conductance-based models, e.g. Equation (3.1), the current flowing through voltage-gated ion channels on the cell membrane is typically modelled as the product of a conductance with a voltage difference (i.e. the current has a linear dependence on membrane potential). What we called the synaptic response, $I(t)$, in the previous section can be thought of as a conductance.

In our model the voltage difference is normalised and so the normalised synaptic current is given by the second term on the right-hand side of equation (3.18),

$$\psi[h(t)]I(t) \quad \text{where} \quad \psi[h(t)] = \frac{h^{eq} - h(t)}{|h^{eq} - h^r|} \quad \text{and} \quad h^{eq} \neq h^r. \tag{3.19}$$

The parameter h^{eq} is referred to as the mean reversal potential, and its value determines whether the synapse is excitatory or inhibitory. If $h^{eq} > h^r$, then $\psi(h^r) = +1$, i.e., the normalised voltage difference is $+1$ when the neuron is at its resting potential. If this is the case, the synapse is excitatory, since synaptic input will generally have a positive effect on the membrane potential.

If $h^{eq} < h^r$, then $\psi(h^r) = -1$, i.e., the normalised voltage difference is -1 when the neuron is at its resting potential. If this is the case, the synapse is inhibitory, since synaptic input will generally have a negative effect on the membrane potential.

Furthermore, if $h = h^{eq}$, synaptic input has no effect on the membrane potential, and the further away h is from h^{eq}, the larger the effect of synaptic input will be.

If $h > h^{eq}$ at an excitatory synapse, then synaptic input will have an inhibitory effect. Similarly, if $h < h^{eq}$ at an inhibitory synapse, then synaptic input will have an excitatory effect.

In the previous section we looked at how the synaptic input influences the mean membrane potential of a neural population. Now we look at how the mean membrane potential influences the synaptic input, through the firing rate.

Our model for the relationship between the mean membrane potential and the population firing rate is somewhat heuristic, but nevertheless physiologically motivated. Within our neural population, there is a distribution of firing thresholds. If the membrane potential of a single neuron is below its firing threshold we assume that its firing rate is 0. If the membrane potential is above the firing threshold for that neuron we assume it is firing at a rate S^{max}.

Increasing the mean membrane potential will increase the proportion of neurons in the population that are above their firing threshold. This proportion is modelled as a sigmoidal function

of the mean membrane potential. The population firing rate is modelled as the product of this sigmoid with the rate S^{max},

$$S(h) = \frac{S^{max}}{1 + \exp\left[-\sqrt{2}(h - \bar{\mu})/\hat{\sigma}\right]}, \tag{3.20}$$

which is qualitatively similar to the cumulative density function of a normal distribution.

The parameters $\bar{\mu}$ and $\hat{\sigma}$ are referred to as the mean firing threshold and the firing threshold standard deviation respectively. The mean firing threshold indicates the mean membrane potential at which half the neural population is firing. The firing threshold standard deviation is a measure of how much variability there is in the firing rate thresholds of the neural population, and also how much variability there is in the membrane potentials of the population.

Now that we have a model for how our firing rate depends on the mean membrane potential, we can write down an equation for the coupling of mean membrane potential dynamics with synaptic input,

$$\begin{aligned} \tau \frac{dh}{dt} &= -(h(t) - h^r) + \psi[h(t)]I(t) \\ \left(\frac{d}{dt} + \gamma\right)^2 I(t) &= NS(h(t)). \end{aligned} \tag{3.21}$$

3.1.4 *Excitatory-inhibitory networks*

We now introduce a simple excitatory-inhibitory network model with three interconnected neural populations, one inhibitory (I), and two excitatory (P and E) populations. This model is linear and we show how it is related to the harmonic oscillator. We then describe the Liley model, which is an extension of the equations introduced in the previous section.

First consider the mean membrane potential dynamics, $h_I(t)$ of neural population I, subject to a synaptic current from excitatory population P,

$$\tau_I \frac{dh_I}{dt} = -h_I + M\ h_P, \tag{3.22}$$

where τ_I is a decay time parameter. The first term on the right-hand side represents exponential decay towards the population resting state. The second term controls whether the population P has an excitatory or an inhibitory effect on the population I. We assume that $M > 0$, meaning that the P to I synapses are excitatory. If M were less than 0 the synapse would be inhibitory.

Now, instead of unidirectional connectivity from P to I assume that current can flow in both directions, and recurrently within each population. Under the same assumptions as above, this leads to a model of the form,

$$\tau_I \frac{dh_I}{dt} = -h_I + M_{PI}\ h_P + M_{II}\ h_I \tag{3.23}$$

$$\tau_P \frac{dh_P}{dt} = -h_P + M_{PP}\ h_P + M_{IP}\ h_I. \tag{3.24}$$

We assume, $\tau_P, \tau_I, M_{PP}, M_{PI} > 0$ and $M_{IP}, M_{II} < 0$.

We can substitute the first equation into the second, using,

$$M_{PI}\ h_P = \tau_I \frac{dh_I}{dt} + (1 - M_{II})\ h_I,$$

and

$$M_{PI}\ \frac{dh_P}{dt} = \tau_I \frac{d^2h_I}{dt^2} + (1 - M_{II})\ \frac{dh_I}{dt}.$$

To make the algebra simpler, assume $M_{PI} = \tau_I = \tau_P$. Then, in terms of the standard parametrization of the harmonic oscillator we have,

$$\frac{d^2h_I}{dt^2} + 2\zeta\omega_0\ \frac{dh_I}{dt} + \omega_0^2\ h_I = 0, \tag{3.25}$$

with,

$$2\zeta\omega_0 = 2 - (M_{PP} + M_{II}),$$

and,

$$\omega_0^2 = -M_{IP} + (1 - M_{PP})(1 - M_{II}).$$

The parameters M_{PP}, M_{II}, M_{IP} and τ_P should be set so that $\omega_0, \zeta > 0$. This is not difficult to achieve, for example, by assuming that $0 < M_{PP} < 1$ and $M_{II} < -1$.

We are now ready to add the third population. Adding a third population extends the range of dynamical behaviour that the model is capable of producing. For example, the power spectrum of resting-state EEG data is characterized by δ-band (0.5-4Hz) oscillations combined with α-band (8-14Hz) oscillations. The two population model in Equations (3.23)-(3.24) cannot reproduce this behaviour. It has also been argued in the literature that three populations consisting of two distinct excitatory populations and an inhibitory population is a better representation of neuronal diversity in cortex, see, for example, [David and Friston, 2003].

The general form of the 3 population model is,

$$\begin{aligned}
\tau_I \frac{dh_I}{dt} &= -h_I + M_{PI}\ h_P + M_{II}\ h_I + M_{EI}\ h_E \\
\tau_P \frac{dh_P}{dt} &= -h_P + M_{PP}\ h_P + M_{IP}\ h_I + M_{EP}\ h_E \\
\tau_E \frac{dh_E}{dt} &= -h_E + M_{PE}\ h_P + M_{IE}\ h_I + M_{EE}\ h_E,
\end{aligned} \tag{3.26}$$

We assume there is no connectivity between the two excitatory populations: $M_{EP}, M_{PE} = 0$. Since E is excitatory, we have $M_{EI}, M_{EE} > 0$, and $M_{IE} < 0$.

By the same method used to derive equation (3.25) we can eliminate population P from the model. If we furthermore add an external input term, $p(t)$, which forces the system away from its stable equilibrium, we obtain the following system of equations,

$$\begin{aligned}
\frac{d^2 h_I}{dt^2} + 2\zeta\omega_0\ \frac{dh_I}{dt} + \omega_0^2\ h_I &= \alpha h_E \\
\frac{dh_E}{dt} &= -\beta h_I + \gamma h_E + p(t),
\end{aligned} \tag{3.27}$$

where $\alpha = M_{EI}$, $\gamma = (M_{EE} - 1)/\tau_E$ and $\beta = -M_{IE}/\tau_E$.

We have derived a model with excitatory and inhibitory interactions between neural populations modelled using linear differential equations. A key feature of the model is that it can produce oscillations in the 8-12Hz frequency range. We will use this simple model for analysis of adult resting-state data in Chapter 7. We now extend our presentation of the Liley *et al* model that we began in Section 3.1.2. Unlike the model above, the Liley *et al* model only has

two physiologically distinct populations. However it is more complex than the model above in the way that it models synaptic conductances. This enables the Liley *et al* to reproduce a wider range of EEG phenomena (including adult resting state data) than the simple three population model above.

The variables in the Liley *et al* model vary with time and space across a 2D cortical sheet due to the effects of long-range excitatory connectivity introduced in Section 3.1.2.

$$\begin{aligned}\tau_e \frac{\partial h_e}{\partial t} &= -(h_e - h_e^r) + \psi_{ee}(h_e) I_{ee} + \psi_{ie}(h_e) I_{ie} \\ \tau_i \frac{\partial h_i}{\partial t} &= -(h_i - h_i^r) + \psi_{ei}(h_i) I_{ei} + \psi_{ii}(h_i) I_{ii}.\end{aligned} \tag{3.28}$$

The state variables are now, $h_e(\mathbf{x}, t)$, the mean membrane potential for the excitatory population; $h_i(\mathbf{x}, t)$, the mean membrane potential for the inhibitory population; and the post-synaptic responses $I_{ee}(\mathbf{x}, t)$, $I_{ei}(\mathbf{x}, t)$, $I_{ie}(\mathbf{x}, t)$, $I_{ii}(\mathbf{x}, t)$.

Each neural population receives two synaptic currents, one excitatory and one inhibitory. The first subscript on I denotes the pre-synaptic population, and the second subscript denotes the post-synaptic population. So, for example, I_{ee} represents recurrent synaptic inputs between neurons in the excitatory population.

The dynamics of the synaptic inputs are now governed by both short-range connectivity, long-range connectivity and extracortical input (e.g. $p_{ee}(\mathbf{x}, t)$). These equations can be written as,

$$\begin{aligned}\left(\frac{\partial}{\partial t} + \gamma_{ee}\right)\left(\frac{\partial}{\partial t} + \tilde{\gamma}_{ee}\right) I_{ee} &= \exp(\gamma_{ee}\delta_{ee})\Gamma_{ee}\tilde{\gamma_{ee}}\left[N_{ee}^{\beta} S_e(h_e) + \Phi_{ee} + p_{ee}(\mathbf{x}, t)\right] \\ \left(\frac{\partial}{\partial t} + \gamma_{ei}\right)\left(\frac{\partial}{\partial t} + \tilde{\gamma}_{ei}\right) I_{ei} &= \exp(\gamma_{ei}\delta_{ei})\Gamma_{ei}\tilde{\gamma_{ei}}\left[N_{ei}^{\beta} S_e(h_e) + \Phi_{ei} + p_{ei}(\mathbf{x}, t)\right] \\ \left(\frac{\partial}{\partial t} + \gamma_{ik}\right)\left(\frac{\partial}{\partial t} + \tilde{\gamma}_{ik}\right) I_{ik} &= \exp(\gamma_{ik}\delta_{ik})\Gamma_{ik}\tilde{\gamma_{ik}}\left[N_{ik}^{\beta} S_i(h_i)\right] \\ \left[\left(\frac{\partial}{\partial t} + v\Lambda\right)^2 - \frac{3}{2}v^2\nabla^2\right]\Phi_{ek} &= v^2\Lambda^2 N_{ek}^{\alpha} S_e(h_e)\end{aligned}$$

$$\Phi_{ik} \equiv 0, \tag{3.29}$$

where $k \in \{i, e\}$, and $\Phi_{ee}(\mathbf{x}, t)$, $\Phi_{ei}(\mathbf{x}, t)$, $\Phi_{ie}(\mathbf{x}, t)$, $\Phi_{ii}(\mathbf{x}, t)$ are additional state variables. A full list of parameters, their definitions, and physiological ranges are provided in Table 3.2.

Note that we now have two shape parameters (γ_{lk} and $\tilde{\gamma}_{lk}$) for each of the four conductance variables. This feature of the model allows greater flexibility for modelling the effect of anaesthetics on PSPs. See Section 3.3.1 for further discussion.

In the previous model, the derivative of the membrane potential depended linearly on the membrane potential of the pre-synaptic population. In this model the relationship is nonlinear, allowing for a much richer repertoire of dynamics, including the behaviour similar to that of the linear model.

Modelling the system as multiple interacting neural populations, as opposed to a single population, increases the size of the parameter space by a factor of 2-4. The resulting model more accurately represents the diversity of neurons present in the human brain. Also, when including the effect of anaesthetics, the model allows for different effects on different types of synapse.

We have also introduced a new set of parameters Γ_{lk}. Adding these parameters effectively adds another parameter to the synaptic response functions, multiplying the response function by a constant. More specifically, the response functions in previous sections assumed that one unit of charge is transferred per pre-synaptic spike, whereas now the charge transferred is, $\Gamma_{lk} \exp(\gamma_{lk}\delta_{lk})/\gamma_{lk}$.

3.2 FIELD POTENTIALS AND THE EEG

The Electroencephalogram (EEG) records a field potential generated by electrical activity in the cortex through an electrode attached to a subject's scalp.

Excitatory synaptic input creates a current sink in the extracellular space at a synapse. (Inhibitory synaptic input creates a current source). Charge is transferred along the dendrite to the soma. Charge leaks from the neuron mainly at the soma (as well as, to a lesser extent, along the dendrite), creating a current source. This dipole in the extracellular electric field can be measured using an electrode. If the spatial distribution of excitatory synaptic input differs from

Table 3.2.: Liley *et al* model parameters

Parameter	Definition	Range	Units
h_e^r, h_i^r	Resting membrane potential	(-80,-60)	mV
τ_e, τ_i	Passive membrane decay time	(5, 150)	ms
h_{ee}^{eq}, h_{ei}^{eq}	Excitatory reversal potential	(-20, 10)	mV
h_{ie}^{eq}, h_{ii}^{eq}	Inhibitory reversal potential	(-90, -65)	mV
Γ_{ee}, Γ_{ei}	Excitatory synaptic input peak amplitude	(0.1, 2.0)	mV
Γ_{ie}, Γ_{ii}	Inhibitory synaptic input peak amplitude	(0.1, 2.0)	mV
δ_{ee}, δ_{ei}	Excitatory PSP rise times to peak	(1, 10)	ms
δ_{ie}, δ_{ii}	Inhibitory PSP rise times to peak	(2, 100)	ms
N_{ee}^{α}, N_{ei}^{α}	Number of excitatory long-range (cortico-cortical) synapses	(2000, 5000)	-
N_{ee}^{β}, N_{ei}^{β}	Number of excitatory short-range (intracortical) synapses	(2000, 5000)	-
N_{ie}^{β}, N_{ii}^{β}	Number of inhibitory short-range (intracortical) synapses	(100, 1000)	-
v	Axonal conduction velocity	(1, 10)	mm/ms
Λ	Decay scale of long-range (cortico-cortical) connectivity	(10, 100)	mm
S_e^{max}, S_i^{max}	Maximum firing rate	(50, 500)	s^{-1}
μ_e, μ_i	Mean firing threshold	(-55, 40)	mV
σ_e, σ_i	Variability in firing thresholds	(2,7)	mV
$\bar{p}_{ee}$, $\bar{p}_{ei}$	Mean rate of extracortical input	(0, 10)	ms^{-1}

the spatial distribution of inhibitory synaptic input, e.g. if excitatory synapses are distributed exclusively at the apex of the cell and inhibitory synapses are distributed exclusively in the basal region, the local field potential will be amplified (see Box 1 in [Pesaran et al., 2018]).

Individual neurons have thousands of dendrites. The morphology of the dendritic tree differs substantially between different types of neuron. The dendrites of inhibitory neurons are arranged in a spherical shape around the soma. This means that when you sum the electric fields generated by synaptic input they will tend to cancel each other out due to symmetry. In contrast the dendrites of excitatory neurons form a tree like structure. the trunks of the dendritic trees are aligned, which results in correlation between the electric fields between neurons. This leads to large dipoles in the net electric field, and creates a field potential that can be recorded at the scalp. The activity of excitatory neurons therefore has a much greater effect on field potentials, such as the EEG, than that of inhibitory neurons. (See Figure 3 in [Lindén et al., 2011]).

In common with other neural population models we assume that the EEG field potential is proportional to the mean membrane potential at the soma, [Nunez and Srinivasan, 1981, Robinson et al., 2001, David et al., 2005, Hutt and Longtin, 2010].

3.3 ANAESTHESIA

3.3.1 *Neurophysiology*

We now model how the post-synaptic response changes in response to the anaesthetic isoflurane. In order to do this, we return to our model for the post-synaptic response to a single pre-synaptic spike, which in Section 3.1.2 was $R_0(t) = \gamma^2 t \exp[-\gamma t]$. The same arguments as above will be applicable for justifying a mean-field approximation for the anaesthesia model.

We would like to model how the post-synaptic response changes in response to changes in isoflurane concentration. In order to make the model more interpretable we parameterize the synaptic response through (i) the rise time, δ, the time taken for the response function, $R(t)$, to reach its maximum, and (ii) the decay time, ζ, the time at which the response function has decayed to some pre-specified value, e.g. ζ satisfies $R(\zeta) = \exp(-1)$ and $\zeta > \delta$.

For the response function from Section 3.1.2, $R_0(t)$, it can be shown, using a Lambert W function, that,

$$\delta = 1/\gamma \quad \text{and} \quad \zeta \approx 3.15\ \delta. \tag{3.30}$$

From this we can see that increasing γ makes both the rise time and the decay time shorter. Experiments looking at the effect of the anaesthetic isoflurane on inhibitory synapses have found that the rise time is not affected by anaesthetic concentration, whereas the decay time lengthens with anaesthetic concentration, [Bojak and Liley, 2005]. In the response model, $R_0(t)$, the decay time cannot be changed without also changing the rise time. In order to overcome this limitation, we introduce the following two parameter response function,

$$R(t) = \gamma\tilde{\gamma}\frac{\exp(-\gamma t) - \exp(-\tilde{\gamma} t)}{\tilde{\gamma} - \gamma}\Theta(t). \tag{3.31}$$

It can be shown that $R(t - t_k)$ is the Green's function for the following differential equation,

$$\left(\frac{d}{dt} + \gamma\right)\left(\frac{d}{dt} + \tilde{\gamma}\right) I(t) = \gamma\tilde{\gamma} N S(t). \tag{3.32}$$

Using the same argument as in the Section 3.1.2, the solution to equation (3.32), when N is large, is a good approximation to the post-synaptic response of a neural population receiving spike-train input from N pre-synaptic neurons firing at an average rate $S(t)$.

For the response function, $R(t)$, the rise time is,

$$\delta = \frac{\epsilon}{\tilde{\gamma} - \gamma} \quad \text{where} \quad \epsilon = \log \frac{\tilde{\gamma}}{\gamma} \tag{3.33}$$

The parameters γ and $\tilde{\gamma}$ uniquely determine δ and ϵ. The decay time, ζ, is also uniquely determined once γ and $\tilde{\gamma}$ are fixed. We can write, $\zeta = f(\gamma, \tilde{\gamma})$. Although we do not have an exact analytic expression for the function f, ζ can be calculated numerically. An approximate analytic expression can be obtained, which is presented and used in Section 7.3-7.4.

In this section, and the thesis more generally we focus primarily on the effect of isoflurane on synaptic receptors. More generally anaesthetic agents can also affect physiology in other ways. For example propofol affects extrasynaptic GABA receptors leading to tonic inhibition [Hutt and Buhry, 2014].

3.3.2 *Macroscopic recordings of brain activity*

The effects of anaesthetics can be observed in the EEG power spectrum. While there are variations in the effect of different anaesthetic agents, the following general observations apply to human EEG and most commonly used anaesthetics, such as propofol, sevoflurane and isoflurane [Kuizenga et al., 2001, John et al., 2001]. During the induction of anaesthesia there is an increase in slow wave activity, also referred to as δ power. This corresponds to activity in the $0 - 4$Hz frequency range of the spectrum. As the anaesthetic concentration increases total power shows a biphasic response, first increasing above the baseline level and then decreasing below the baseline level.

The Liley *at al* model predicts these changes in the EEG for selected parameter sets within physiological ranges [Bojak and Liley, 2005]. The parameter sets that reproduced these ob-

servations exhibited approximately linear dynamics around a stable fixed point, leading to a description of the EEG signal as linearly filtered thalamic input.

At deep concentrations of anaesthesia bursting is observed, i.e., short periods of high amplitude activity punctuate otherwise relatively small fluctuations in neural activity [Rampil, 1998]. At even deeper levels of anaesthesia, bursting disappears, and the amplitude of fluctuations continues to decrease towards zero.

Burst suppression is predicted by an extension of the Liley *et al* model that accounts for synaptic resource depletion [Bojak et al., 2015]. The timescale of synaptic resource depletion is slower than that of the neural dynamics leading to transitions between stable and unstable system dynamics for parameter sets that characterize the deep levels of anaesthesia where burst suppression is observed.

Somewhat similar changes in the EEG power spectrum can be observed in rats anaesthetized with isoflurane, something we will return to in Chapter 7.

3.4 INFERENCE FOR NEURAL POPULATION MODELS

Suppose that we want to make predictions using a Neural Population Model (NPM). For example we may want to predict the effect of anaesthetic agents on the spectrum of the EEG signal. Two basic requirements are that the model is an accurate representation of neural activity and that simulations from the model are tractable. If, in addition, we are able to obtain independent and accurate measurements of all the parameters in the model as a function of anaesthetic concentration then we are in a position to make predictions.

In reality we may not be able to measure individual parameters directly. However it may still be possible to calibrate the model. I.e. given some observations, e.g. EEG time-series, we can find maximum likelihood parameters (or MAP parameters in a Bayesian setting) that best explain the data. These point estimates of the model parameters can then be used to make predictions. However the accuracy of these predictions depends on the level of uncertainty in estimates of the model parameters (and on the accuracy of the model itself).

Within Computational Neuroscience the most systematic and extensive methodology for quantifying uncertainty in NPMs is Dynamic Causal Modelling (DCM). This literature uses several approximations, for example the variational Laplace approximation, which assumes that uncertainty in parameters can be modelled by a Gaussian, and furthermore that certain sets of parameters are independent of each other. This assumption does seem to be justified for some specific models [Friston et al., 2007]. However there are no guarantees that this assumption holds in general, it can only really be checked a problem by problem basis by benchmarking against more accurate (and often more computationally expensive) methods. As Computational Neuroscience models become more complex (and thus more expensive to simulate and analyze) this assumption is less likely to hold. As we will see in Chapters 5 and 7 posterior distributions for the Liley *et al* model and even for the relatively simple three population model in Section 3.1.4 can be highly non-Gaussian due to a lack of identifiability and the likelihood being a nonlinear function of the model parameters. This motivates the use of MCMC methods for sampling the posterior distribution. MCMC methods have seen some use in the DCM and wider Computational Neuroscience literature [Penny and Sengupta, 2016, Abeysuriya and Robinson, 2016, Hashemi et al., 2018]. However their widespread adoption depends on further advances in computational efficiency. The main contribution of Chapter 5 will be the development of a novel method that improves the computational efficiency of MCMC with applications to the case of inferring NPM parameters from stationary EEG time-series.

Two further approximations that are widely used in the DCM literature for steady-state response models are linearization around a stable equilibrium and an asymptotic approximation for the likelihood in the spectral domain [Moran et al., 2009]. These approximations can be avoided through use of a particle filter. Again the adoption of particle filters in Computational Neuroscience is limited because of the prohibitive computational cost. Improving the efficiency of particle filters is challenging and remains an open problem. The main contribution of Chapter

4 is to develop methods to further quantify the accuracy of the linearization and asymptotic approximation that are used in spectral data analysis in Computational Neuroscience.

4

STATIONARY PROCESSES AND THE WHITTLE LIKELIHOOD

There have been several recent papers in the Computational Neuroscience literature on methods for estimating parameters of Neural Population Models (NPMs) from stationary time-series data [Moran et al., 2009, Abeysuriya and Robinson, 2016, Hashemi et al., 2018]. These methods compare spectral properties of the data with a prediction of the spectral density from the model, and typically linearize the model around a stable fixed point, which enables rapid prediction of the spectral density, thus assuming that the differences between the spectral density of the original model and the linearized model are negligible. Furthermore, they assume that the asymptotic error that arises from working in the spectral domain is negligible. When are these assumptions justified? That is the question we set out to answer in this chapter. While there are some qualitative answers to these questions in the existing literature, to our knowledge, the study here is novel in offering more quantitative answers. We identify scenarios where the assumptions do not hold and develop methodology that could be used to quantify these errors for other similar problems.

The structure of the chapter is as follows. Sections 4.1 and 4.2 review existing literature on continuous-time stochastic processes and spectral analysis, including the Whittle likelihood. Knowledge of this literature is relatively sparse among the statisticians and neuroscientists, with the majority of people I have spoken to not having heard of the Whittle likelihood. So one of the contributions of this chapter is to make these results more accessible to other researchers.

Section 4.3 reviews a generally applicable approach for computing the spectral density of linear state-space models, and also contains novel material in the derivation of analytic derivatives of the spectral density. In Section 4.4 we evaluate the accuracy of the Whittle likelihood by comparing it with the particle filter and Kalman filter likelihoods that were reviewed in Section 2.6.

4.1 PRELIMINARIES

In general, a model for time-series data will consist of two components: the deterministic part and the random part. Purely deterministic models are often referred to as dynamical systems, and the more general case may be referred to as noise-driven dynamical systems.

4.1.1 *Stationary Processes*

Informally, a process is stationary if when we take any realization of the process and divide it into a number of time intervals, the various sections of the realizations look "pretty much" the same [Priestley, 1981, p. 14].

More formally, the process $X(t)$ is said to be completely stationary if, for any admissible $t_1, t_2, \ldots, t_n$, and any k, the joint probability distribution of $\{X(t_1), X(t_2), \ldots, X(t_n)\}$ is identical with the joint probability distribution of,

$$\{X(t_1 + k), X(t_2 + k), \ldots, X(t_n + k)\}. \tag{4.1}$$

The process $X(t)$ is said to be stationary up to order m if, for any admissible $t_1, t_2, \ldots, t_n$, and any k, all the joint moments up to order m, of $\{X(t_1), X(t_2), \ldots, X(t_n)\}$ exist and equal the corresponding joint moments up to order m of $\{X(t_1 + k), X(t_2 + k), \ldots, X(t_n + k)\}$ [Priestley, 1981, p. 106].

The second moment of the process $X(t)$ is referred to as the autocovariance. In general, the autocovariance is a function of time,

$$\mu = \mathrm{E}[X(t)] \tag{4.2}$$

$$\mathrm{Cov}[X(t), X(t+\tau)] = \mathrm{E}[(X(t) - \mu)(X(t+\tau) - \mu)] \tag{4.3}$$

For stationary processes we may write the autocovariance as a function of τ,

$$R(\tau) = \mathrm{E}[(X(t) - \mu)(X(t+\tau) - \mu)] \tag{4.4}$$

4.1.2 *The Wiener Process and the damped harmonic oscillator*

An example of a dynamical system, that we will use as a running example is the damped harmonic, oscillator,

$$\frac{d^2}{dt^2}X(t) + \alpha_1 \frac{d}{dt}X(t) + \alpha_2 X(t) = \epsilon(t) \tag{4.5}$$

This equation can be used to describe a pendulum swinging in a resistance medium. The term on the right-hand side, $\epsilon(t)$ is described as the driving force. If $\epsilon(t)$ is zero, and the parameters α_1, α_2 are such that the system is stable, the state variable $X(t)$ will tend towards zero, which corresponds to the pendulum being at rest.

The harmonic oscillator is noise-driven if $\epsilon(t)$ is a random process. In 1927, Yule published a paper where he imagined that $\epsilon(t)$ might describe the effect of a boy with a pea-shooter shooting peas at the pendulum at random time instants [Priestley, 1981, p. 132]. In Chapter 3 we considered more complex examples of dynamical systems that describe the response of neurons in the brain to incoming electrical signals. In that case $\epsilon(t)$ describes random fluctuations in neuronal firing rates.

One of the most basic and commonly used examples of a random process is the Wiener process. The continuous-time process $W(t)$ is a Wiener process if the increment $W(t) - W(s)$ is normally distributed with mean 0 and variance $\sigma_W^2(t-s)$, and increments for non-overlapping time intervals are independent [Weisstein, 2018].

The continuous AR(2) process (i.e. continuous-time second-order auto-regressive process) is obtained by setting $\epsilon(t) = dW(t)$. The expression $dW(t)$ is, loosely speaking, the derivative of the Wiener process and is equivalent to a white noise process, i.e. a process where the autocorrelation is zero for non-zero increments. To give $dW(t)$ a well-defined meaning suppose that,

$$\epsilon_h(t) = \frac{W(t+h) - W(t)}{h} \tag{4.6}$$

The autocovariance of $\epsilon_h(t)$ is [Priestley, 1981, p. 161],

$$R_h(\tau) = \begin{cases} (\sigma_W^2/h)(1 - |\tau|/h) & \text{if } |\tau| < h, \\ 0 & \text{if } |\tau| \geq h. \end{cases} \tag{4.7}$$

Taking the limit of $R_h(\tau)$ as $h \to 0$ tells us that the autocovariance of $dW(t)$ is,

$$R(\tau) = \sigma_W^2 \delta(\tau), \tag{4.8}$$

where $\delta(\tau)$ is the Dirac delta function.

The Wiener process is an example of a *non*-stationary process. It can be shown [Priestley, 1981, p. 162] that,

$$\text{Var}[W(t)] = t\,\sigma_w. \tag{4.9}$$

In other words, the second moment of $W(t_0)$ is not equal to the second moment of $W(t_0 + k)$ for any choice of t_0, and non-zero k.

The continuous AR(2) process can be either stationary or non-stationary depending on the values of α_1 and α_2. The deterministic part of the model (a homogeneous ODE) has a fixed point attractor at $X(t) = 0$. If this fixed point is stable, the AR(2) process is stationary, and if it is unstable, the AR(2) process is non-stationary.

We can evaluate the stability of the fixed point by writing the model as a system of first-order homogeneous ODEs,

$$\begin{bmatrix} \dot{x} \\ \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \\ -\alpha_2 & -\alpha_1 \end{bmatrix} \begin{bmatrix} x \\ \\ z \end{bmatrix} \tag{4.10}$$

The eigenvalues of the Jacobian are given by,

$$\lambda_1 = \frac{1}{2}\Big(-\alpha_1 - \sqrt{\alpha_1^2 - 4\alpha_2}\Big) \tag{4.11}$$

$$\lambda_2 = \frac{1}{2}\Big(-\alpha_1 + \sqrt{\alpha_1^2 - 4\alpha_2}\Big) \tag{4.12}$$

The system is stable if both eigenvalues have negative real parts. This condition is satisfied when both $\alpha_1, \alpha_2 > 0$.

The solution to the homogeneous ODE can be written as [Priestley, 1981, p. 171],

$$x(t) = A_1 \exp(\lambda_1 t) + A_2 \exp(\lambda_2 t), \tag{4.13}$$

where A_1, A_2 are arbitrary constants, which can be determined by setting the initial conditions, $x(0), \dot{x}(0)$.

It is clear from (4.13) that if $\mathrm{Re}[\lambda_1] > 0$ or $\mathrm{Re}[\lambda_2] > 0$, the magnitude of solutions to the model will grow exponentially over time (unless $A_1, A_2 = 0$). Therefore, in this case, the corresponding AR(2) process is non-stationary.

When $\alpha_1, \alpha_2 > 0$ it can be shown that the autocorrelation function is [Priestley, 1981, p. 172],

$$\rho(\tau) = \frac{\lambda_2 \exp(\lambda_1|\tau|) + \lambda_1 \exp(\lambda_2|\tau|)}{\lambda_2 - \lambda_1}. \tag{4.14}$$

As this only depends on the interval between two time-points, τ, this demonstrates that the process is stationary up to order 2.

In the case where the roots are complex, i.e., when $\alpha_2 > \alpha_1^2/4$, we have,

$$\exp(\lambda_1|\tau|) = \exp\left(-\frac{1}{2}\alpha_1|\tau|\right)\left(\cos(\phi|\tau|) + i\sin(\phi|\tau|)\right) \tag{4.15}$$

where $\phi = \sqrt{\alpha_2 - \alpha_1^2/4}$. Equation (4.14) can then be re-written as,

$$\rho(\tau) = \exp\left(-\frac{1}{2}\alpha_1|\tau|\right)\left(\cos(\phi|\tau|) + \frac{2\phi}{\alpha_1}\sin(\phi|\tau|)\right). \tag{4.16}$$

In other words, $\rho(\tau)$ decays sinusoidally with 'period' $2\pi/\phi$.

4.1.3 *Linearization and approximately stationary processes*

A general form for many continuous-time nonlinear dynamical systems is

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}[\mathbf{x}(t);\theta] + \mathbf{p}(t;\theta), \tag{4.17}$$

where $\mathbf{x}$ is a vector of states with length d, θ is a vector of parameters, and $\mathbf{F}$ is a vector-valued nonlinear function, also of length d. This form separates the time-invariant "system" given by $\mathbf{F}$ and θ from an explicitly time-dependent "input" $\mathbf{p}$. Aspects of many natural and engineered systems are approximated well by this form, due to large differences in the time-scales of other processes. For example, brain connectivity changes considerably over a human lifetime, but typically can be considered as a static structure for the description of neural processes lasting milliseconds to seconds.

If $\mathbf{p}(t;\theta)$ is a deterministic function, then Equation (4.17) represents a system of coupled Ordinary Differential Equations (ODEs). This is particularly appropriate if we model a stimulus (a distinct event, or series of events) through the input $\mathbf{p}(t;\theta)$. We then assume that $\mathbf{p}(t;\theta)$ is zero as default, but becomes non-zero in a characteristic way at times when the system receives a stimulus. If $\mathbf{p}(t;\theta)$ is the time derivative of a multivariate Wiener process, then Equation (4.17) represents a multivariate diffusion process with drift term $\mathbf{F}[\mathbf{x};\theta]dt$. The conventional notation for diffusion processes can be obtained by multiplying through by dt, and setting $\mathbf{p}(t;\theta)dt = \boldsymbol{\Sigma}(\theta)d\mathbf{W}_t$, with $\boldsymbol{\Sigma}(\theta)$ a diagonal matrix, and $\mathbf{W}_t$ a standard uncorrelated multivariate Wiener process. We assume that $\mathbf{p}(t;\theta)$ has mean zero. Such an SDE is particularly appropriate for "resting state" conditions, where the system is not primarily driven by sepa-

rable and characteristic events (at least to our knowledge), but noisy internal and/or external fluctuations dominate.

A further distinction that is key in the context of stationary processes is between what we refer to as stable and unstable systems. To understand these labels, we first have to discuss the so-called equilibrium or fixed points of a system. A point $\mathbf{x}^*$ is an equilibrium point of the system dynamics if its state remains constant in this point: $d\mathbf{x}/dt|_{\mathbf{x}=\mathbf{x}^*} = 0$. To be more specific, we consider here equilibrium points for "least input" scenarios: in the absence of all stimuli (deterministic case) or for exactly average input (stochastic case), respectively. In both cases, we would set $\mathbf{p}(t;\theta) \equiv 0$ in Equation (4.17) due to our definition of the input term above. The equilibria are then solutions of $\mathbf{F}[\mathbf{x}^*;\theta] = 0$. The stability of an equilibrium point is evaluated by examining the eigenvalues of $\mathcal{J}$, the Jacobian matrix of $\mathbf{F}$ evaluated for $(\mathbf{x}^*;\theta)$. The condition for stability of an equilibrium point is

$$\max_{n=1,\ldots,d}(\mathrm{Re}[\lambda_n]) < 0, \tag{4.18}$$

where $\lambda_1, \ldots, \lambda_d$ are the eigenvalues of $\mathcal{J}(\mathbf{x}^*;\theta)$. The Jacobian is calculated by taking the partial derivatives of $\mathbf{F}$ with respect to the state variables

$$[\mathcal{J}]_{ij}(\mathbf{x}^*,\theta) = \left.\frac{\partial[\mathbf{F}]_i(\mathbf{x};\theta)}{\partial[\mathbf{x}]_j}\right|_{\mathbf{x}=\mathbf{x}^*}. \tag{4.19}$$

In the following we will be concerned with methods for systems evaluated near such a stable equilibrium. We will loosely call a system "stable" if such a description makes sense at all. Thus we will refer to a system as stable if it possesses at least one stable equilibrium point to which these methods apply. There may be some regions of the parameter space where the system has no stable equilibrium points. However, provided that one can find a region in the available parameter space where it does, then we will still call this a "stable" case – with the implicit understanding that any parameter inference with our methods must concern the system operating in the basin of a stable fixed point attractor.

Spectral analysis using the Whittle likelihood is applicable to stable SDEs. The methodology we present for rapidly calculating the spectral density requires the model to be linearized. It is therefore only applicable when the input is sufficiently small for the linearized model to be a good approximation to the nonlinear model. A classic example is the description of a pendulum as harmonic oscillator, which is valid as long the angle of its swings are small enough for $\sin\phi \approx \phi$.

The dynamics of mesoscopic brain oscillations in the resting state, as measured for example with the electroencephalogram (EEG), are a prominent case in point [van Rotterdam et al., 1982, Liley et al., 2002, Breakspear and Terry, 2002]. Generally speaking, many nonlinear systems have operating regimes of interest where the system dynamics are thought to be (or are designed to be) quasi-linear, as guaranteed by the local stable manifold theorem [Kelley, 1967].

Note that linearizing around an equilibrium point is different from repeatedly linearizing the model around a sequence of distinct points in the state-space. In general linearizing around an equilibrium point will be more approximate because the linearization is only accurate when the system is close to its equilibrium state. To assess the accuracy of this approximation for a given set of parameter values, one can compare simulated data from the nonlinear model and from the linearized model. An example of this type of analysis can be found in Section IV of [Bojak and Liley, 2005]. Assessing the effect of linearization on posterior inference is somewhat more challenging, and we come back to this in Section 4.4.

4.1.4 *FitzHugh-Nagumo equations*

The FitzHugh-Nagumo (FHN) model was briefly introduced in Section 3.1.1. We now give a slightly more detailed discussion of the model highlighting aspects that are useful for testing the methodology we develop in this chapter and in the next chapter.

In Section 4.4, we study a stochastic version of the FHN model in order to compare the use of the time-domain likelihoods described in Section 2.6, namely the particle filter and the extended Kalman filter against a Kalman filter with a time-homogeneous linearization of FHN about a stable equilibrium point.

There are parameter sets for which the FHN model has a stable equilibrium point, and where, given a sufficiently large perturbation away from the stable equilibrium, the model produces a nonlinear transient oscillation. There are also parameter sets where a sufficiently large perturbation from stable equilibrium can cause the system to move towards a limit-cycle attractor. However, we do not consider that case here. The FHN equations that we will use here are

$$\frac{d}{dt}V(t) = V(t)[a - V(t)][V(t) - 1] - w(t) + I_0 + I(t) \tag{4.20}$$

$$\frac{d}{dt}w(t) = bV(t) - cw(t) + d + P(t). \tag{4.21}$$

The state variables V and w are referred to as the membrane potential and the recovery variable, respectively, when the model is used in the neuroscience domain. Compared to the standard FHN form [Izhikevich, 2007], we have added here terms d and $P(t)$ to the recovery variable, to allow additional analysis below. We consider $I(t)$ here as zero by default / mean-zero, and hence in comparison to the standard notation have explicitly separated out a possible constant input as I_0. As discussed in Section 4.1.3, we intend $P(t)$ to be zero by default in the ODE and mean-zero in the SDE case, with d parametrising any constant perturbation. However, this is not intended as a new model proposal in the domain science, and obviously the standard form will be recovered if these terms are set to zero.

Here and in the following we always consider equilibria for default inputs $P(t) = I(t) = 0$. The steady-state equations, also known as nullclines, are then obtained by setting the derivatives dV/dt and dw/dt to zero. If constant inputs d and I_0 are absent as well, then it is easy to see from Equations (4.20) and (4.21) that there is an equilibrium at $V^* = w^* = 0$. As shown, equilibrium values of state variables will be indicated by a star superscript. This particular equilibrium will be stable if $a + c > 0$ and $ac + b > 0$, as follows from computing the eigenvalues of the Jacobian:

$$\mathcal{J} = \begin{pmatrix} -a & -1 \\ b & -c \end{pmatrix}. \tag{4.22}$$

4.2 SPECTRAL ANALYSIS FOR UNIVARIATE PROCESSES

In order to define the spectral density it is convenient to truncate the stationary process, X(t), to the interval $[-T, T]$,

$$X_T(t) = \begin{cases} X(t) & \text{when } t \in [-T, T] \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

The Fourier Transform of the truncated stationary process, $X_T(t)$, is defined as,

$$\tilde{X}_T(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_T(t) \exp[-i\omega t] dt. \tag{4.24}$$

The power of $X(t)$, at frequency ω is defined as,

$$\tilde{X}(\omega) = \lim_{T \to \infty} \frac{|\tilde{X}_T(\omega)|^2}{2T}. \tag{4.25}$$

The quantity $|\tilde{X}_T(\omega)|^2$ can be thought of as the total energy of $X(t)$ at frequency ω over the interval $[-T, T]$, and the power is then simply energy per unit time.

The (non-normalized) power spectral density function, or simply spectral density, of $X(t)$, is the average power at ω over all realizations of $X(t)$,

$$f(\omega) = \lim_{T \to \infty} \mathrm{E}\left[\frac{|\tilde{X}_T(\omega)|^2}{2T}\right] \tag{4.26}$$

The basic idea of inference in the frequency domain is to model the probability distribution of Fourier-transformed data using the spectral density. In order to implement this approach we need to be able to evaluate the spectral density of $X(t)$.

Assuming a weakly stationary mean-zero process, the Wiener-Khinchin theorem states that the spectral density is the Fourier Transform of the autocovariance function,

$$f(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp[-i\omega t] R(\tau) d\tau. \tag{4.27}$$

The proof of this is somewhat lengthy (albeit not particularly complex), [Priestley, 1981, p. 210-213]. A key intermediate result is that,

$$|\tilde{X}_T(\omega)|^2 = \int_{-\infty}^{\infty} \exp[-i\omega t]k(t)dt, \tag{4.28}$$

where $k(t)$ is the convolution of $X_T(\omega)$ with itself,

$$k(\tau) = \int_{-\infty}^{\infty} \frac{X_T(u)}{\sqrt{2\pi}} \cdot \frac{X_T(u-\tau)}{\sqrt{2\pi}} du \tag{4.29}$$

The basic idea of the proof is to take the expected value of both sides of Equation (4.28) and analyze the $T \to \infty$ limit.

For some stationary processes it is possible to derive the spectral density analytically by taking the Fourier Transform of the autocovariance function. For example, it can be shown that for the harmonic oscillator [Priestley, 1981, p. 239],

$$f_X(\omega) = \frac{1}{\pi}\left(\frac{\alpha_1\alpha_2}{(\alpha_2-\omega)^2 + \alpha_1^2\omega^2}\right) \cdot \sigma_\epsilon^2, \tag{4.30}$$

where σ_ϵ^2 is the variance of the white noise process $\epsilon(t)$.

Alternatively, observe that the Fourier Transform of $dX_T(t)/dt$ is

$$\int_{-\infty}^{\infty} \frac{dX_T(t)}{dt} \exp[-i\omega t]dt = \Big[X_T(t)\exp[i\omega t]\Big]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} X_T(t)(-i\omega)\exp[-i\omega t]dt \tag{4.31}$$

$$= i\omega\tilde{X}_T(\omega) \tag{4.32}$$

Applying the Fourier Transform to both sides of the harmonic oscillator equation we get,

$$(i\omega)^2\tilde{X}_T(\omega) + \alpha_1(i\omega)\tilde{X}_T(\omega) + \alpha_2\tilde{X}_T(\omega) = \tilde{\epsilon}_T(\omega) \tag{4.33}$$

$$\implies \qquad |\alpha_2 - \omega^2 + \alpha_1\omega i|^2|\tilde{X}_T(\omega)|^2 = |\tilde{\epsilon}_T(\omega)|^2 \tag{4.34}$$

By taking expectations of both sides, and the limit as $T \to \infty$, then re-arranging we can obtain the same equation for the harmonic oscillator as in Equation (4.30). As we will see in Section 4.3, this second method can be applied to linear systems, and is generally useful in settings where an analytic expression for the autocovariance function is not available.

4.2.1 *Sums of stationary processes*

Typically one assumes that the difference between the (discretely sampled) process and the observations is a stochastic process, such as white noise. When two processes $X(t)$ and $Z(t)$ are independent, then the autocovariance of $Y(t) = X(t) + Z(t)$ is simply $R_Y(\tau) = R_X(\tau) + R_Z(\tau)$. It then follows from Equation (4.27) that $f_Y(\omega) = f_X(\omega) + f_Z(\omega)$. If $Z(t)$ is zero-mean white noise, then $R_Z(\tau) = \sigma_Z^2 \delta(\tau)$ and consequently the measured spectral density will have a "noise floor" $f_Y(\omega) = f_X(\omega) + \sigma_Z^2$.

4.2.2 *Discrete-time observations of continuous-time processes*

Suppose that we observe the continuous-time process, $X(t)$, at a discrete set of time points,

$$Y_t = X(t \cdot \Delta t) \tag{4.35}$$

for integer $t \geq 0$ and Δt some fixed observation time-step. Whereas the spectral density of $X(t)$ is defined for all real-valued $\omega \in (-\infty, \infty)$, the spectral density of Y_t is restricted to the interval $(\frac{-\pi}{\Delta t}, \frac{\pi}{\Delta t})$.

The spectral density at frequencies outside $(\frac{-\pi}{\Delta t}, \frac{\pi}{\Delta t})$ is "folded in" to $(\frac{-\pi}{\Delta t}, \frac{\pi}{\Delta t})$, an effect that is referred to as aliasing [Priestley, 1981, p. 504],

$$f_Y(\omega) = \sum_{k=-\infty}^{k=\infty} f_X\left(\omega + \frac{2k\pi}{\Delta t}\right), \quad |\omega| \leq \frac{\pi}{\Delta t}. \tag{4.36}$$

It is therefore desirable to observe the system with a time resolution that minimizes the aliasing effect i.e. such that the spectral density is effectively zero outside of $(\frac{-\pi}{\Delta t}, \frac{\pi}{\Delta t})$. In that case $f_Y(\omega) \approx f_X(\omega)$.

We may want to change the time units in our model. Let $t' = \alpha t$ be a transformation of the timescale, with t' representing time on the transformed units. For example, if $\alpha = 1/\Delta t$, the transformed time-points are integers. If the original time units were seconds, and $\alpha = 10^3$, the transformed time-units are milliseconds.

Changing the time units changes the interval on which the spectral density of Y_t is defined to $(\frac{-\pi}{\alpha\Delta t}, \frac{\pi}{\alpha\Delta t})$. The spectral density under the new time units can be defined in terms of the original spectral density as [Priestley, 1981, p. 509],

$$f_Y^*(\omega') = \alpha f_Y(\omega), \quad \text{where } \omega' = \omega/\alpha. \tag{4.37}$$

The proof of this relationship is somewhat involved and requires an understanding of stochastic calculus.

4.2.3 *Whittle likelihood*

In this section we briefly recall the derivation of the Whittle likelihood, following [Shumway and Stoffer, 2011]. We will refer back to this in Section 4.4.2, where we study the accuracy of this approximation. We assume that the process has zero mean, but it is not difficult to generalize the Whittle likelihood to the non-zero case. The periodogram of a time series is the squared modulus of its Discrete Fourier Transform (DFT),

$$\begin{aligned} S_k = \tilde{Y}_k^2 &= \sum_l Y_l e^{-2\pi i k l/n} \\ &= \left[\frac{1}{\sqrt{n}} \sum_l Y_l \cos\frac{2\pi k l}{n}\right]^2 + \left[\frac{1}{\sqrt{n}} \sum_l Y_l \sin\frac{2\pi k l}{n}\right]^2 \\ &= {Y_k^c}^2 + {Y_k^s}^2. \end{aligned} \tag{4.38}$$

with Y^c and Y^s the discrete cosine and sine transforms. If the time series $\{Y_l\}$ is a normal random variable with mean zero then, as linear functions of $\{Y_l\}$, Y_k^c and Y_k^s will also be jointly normal with mean zero. If we transform the units of the spectral density using $\alpha = 1/\Delta t$, the variances and covariances are as follows, [Shumway and Stoffer, 2011]:

$$\mathrm{Cov}[Y_k^c, Y_l^c] = \frac{f_Y(\omega_k)}{2}\delta_{kl} + \epsilon_n \tag{4.39}$$

$$\mathrm{Cov}[Y_k^s, Y_l^s] = \frac{f_Y(\omega_k)}{2}\delta_{kl} + \epsilon_n \tag{4.40}$$

$$\mathrm{Cov}[Y_k^c, Y_l^s] = \epsilon_n, \tag{4.41}$$

where $f_Y(\omega_k)$ is the spectral density of the discrete-time process Y_t, $\omega = \frac{k}{n} \cdot \pi$, and δ_{kl} is a Kronecker delta. The variable ϵ_n depends on k, l, c and s as well as n, but it can be bounded by an expression that depends only on n,

$$|\epsilon_n| < \phi/n \text{ with } \phi = \sum_{m=-\infty}^{\infty} |m| \, |R(m \cdot \Delta t)| \,, \tag{4.42}$$

where $R(\tau)$ is the autocovariance function introduced above.

The Whittle likelihood is obtained by dropping the ϵ_n terms. The sum of two squared standard normal random variables is a χ^2 distribution with two degrees of freedom. From this we can deduce that the probability density of $S_k = {Y_k^c}^2 + {Y_k^s}^2$ in Equation (4.38) is, $\frac{1}{f_Y(\omega_k)} \exp\left[-\frac{S_k}{f_Y(\omega_k)}\right]$. And since S_k is approximately independent of S_l for $k \neq l$ we have,

$$p_\theta(y_0, \ldots, y_{n-1}) = p_\theta(S_0, \ldots, S_{n-1}) \approx \prod_{k=1}^{n/2-1} \frac{1}{f_Y(\omega_k)} \exp\left[-\frac{S_k}{f_Y(\omega_k)}\right], \tag{4.43}$$

where we have, as is common, neglected the mean and Nyquist spectral edges $k = 0, n/2$. Note that the terms $S_{n/2}, \ldots, S_{n-1}$ do not appear in the product since they are fully determined by $S_1, \ldots, S_{n/2-1}$ for real signals. The equation $|\epsilon_n| < \phi/n$ tells us that the error in the approximation decreases linearly with n, and also that it depends on the decay rate of the autocovariance function. The effect of the error on inference, and how this varies with ϕ and n is analysed in Section 4.4.2.

Note that the Whittle likelihood is invariant (up to a multiplicative constant) to the unit of frequency. Here we have chosen to use the standard units for discrete time processes, but we could have used the original time units in our model, in which case the periodogram would need to be multiplied by Δt.

In certain situations it is possible to remove the bias in the Whittle likelihood that arises due to aliasing and finite sample size effects. See [Sykulski et al., 2016] for more details.

4.3 SPECTRAL ANALYSIS FOR LINEAR SYSTEMS

Any system of linear (stochastic) differential equations can be written in the form

$$\frac{d}{dt}\mathbf{X}(t) = A\,\mathbf{X}(t) + \mathbf{P}(t). \tag{4.44}$$

Truncating $\mathbf{X}(t)$ and applying a Fourier transform to both sides of Equation (4.44) we obtain

$$(i\omega I - A)\,\tilde{\mathbf{X}}_T(\omega) = \tilde{\mathbf{P}}_T(\omega). \tag{4.45}$$

For a given value of ω, Equation (4.45) is a linear system of equations, which can be re-arranged to obtain $\tilde{\mathbf{X}}_T(\omega)$ as a function of $\tilde{\mathbf{P}}_T(\omega)$. The matrix $\mathcal{T}(\omega)$ that maps $\tilde{\mathbf{P}}_T(\omega)$ to $\tilde{\mathbf{X}}_T(\omega)$ is called the transfer function

$$\tilde{\mathbf{X}}_T(\omega) = \mathcal{T}(\omega)\ \tilde{\mathbf{P}}_T(\omega). \tag{4.46}$$

If we divide through by $2T$ and take the squared absolute value of both sides we get,

$$\frac{|\tilde{\mathbf{X}}_T(\omega)|^2}{2T} = \frac{|\mathcal{T}(\omega)\ \tilde{\mathbf{P}}_T(\omega)|^2}{2T} \tag{4.47}$$

If $\mathbf{P}(t)$ is such that it only contains one time-varying component, $P_j(t)$, then we can easily obtain an expression in terms of the spectral densities,

$$f_{X_i}(\omega) = |\mathcal{T}_{ij}(\omega)|^2\ f_{P_j}(\omega), \tag{4.48}$$

where $f_{P_j}(\omega)$ is the spectral density of $P_j(t)$, and f_{X_i} is the spectral density of the ith component of $\mathbf{X}$.

This formula can be generalized to cases where there is more than one time-varying component. For simplicity we consider the case of two time-varying components. Similar arguments can be used for any number of time-varying components. We temporarily drop the subscript T that

represents truncation of the process and use the subscript here to denote entries of $\mathcal{T}$ and $\tilde{\mathbf{P}}$. And we drop the ω argument, which remains implicit.

$$|\mathcal{T}_{ij}\ \tilde{P}_j + \mathcal{T}_{ik}\ \tilde{P}_k|^2 = (\mathcal{T}_{ij}\ \tilde{P}_j + \mathcal{T}_{ik}\ \tilde{P}_k)\,(\mathcal{T}_{ij}\ \tilde{P}_j + \mathcal{T}_{ik}\ \tilde{P}_k)^* \tag{4.49}$$

$$= (\mathcal{T}_{ij}\ \tilde{P}_j + \mathcal{T}_{ik}\ \tilde{P}_k)\left(\mathcal{T}^*_{ij}\ \tilde{P}^*_j + \mathcal{T}^*_{ik}\ \tilde{P}^*_k\right) \tag{4.50}$$

$$= |T_{ij}|^2|P_j|^2 + |T_{ik}|^2|P_k|^2 + \mathcal{T}_{ij}\mathcal{T}^*_{ik}\ \tilde{P}_j\tilde{P}^*_k + \mathcal{T}^*_{ij}\mathcal{T}_{ik}\ \tilde{P}^*_j\tilde{P}_k \tag{4.51}$$

When there are two time-varying inputs, the spectral density of $X(t)$ is then,

$$f_{X_i}(\omega) = |\mathcal{T}_{ij}(\omega)|^2\ f_{P_j}(\omega) + |\mathcal{T}_{ik}(\omega)|^2\ f_{P_k}(\omega) + (\mathcal{T}_{ij}\mathcal{T}^*_{ik} + \mathcal{T}^*_{ij}\mathcal{T}_{ik}) f_{P_jP_k}(\omega), \tag{4.52}$$

where $f_{P_jP_k}(\omega)$ is the cross-spectral density of P_j and P_k. If P_j andP_k are independent, the cross spectral density, and hence the last term is zero.

An expression for $\mathcal{T}(\omega)$ can be obtained using the eigen-decomposition of A [Bojak and Liley, 2005]:

$$\mathcal{T}(\omega) = \mathcal{R}\ \mathrm{diag}\left[\frac{1}{e_k(i\omega - \lambda_k)}\right]\mathcal{L}, \tag{4.53}$$

where the j-th column of $\mathcal{R}$ is the j-th right-eigenvector of A, the i-th row of $\mathcal{L}$ is the i-th left-eigenvector, $\lambda_1, \ldots, \lambda_k, \ldots, \lambda_d$ are its eigenvalues, and $e_1, \ldots, e_k, \ldots, e_d$ are norms of the eigenvectors obtained from $\mathcal{L}\mathcal{R} = \mathrm{diag}[e_k]$.

This result provides us with an important observation: the computational complexity of evaluating the spectral density can be reduced if only some of the elements of the transfer function matrix are needed. When $P_j(t)$ is the only time-varying input we only ever require the jth column of $\mathcal{T}(\omega)$. If we only observe one element of $\mathbf{X}(t)$ we only require the ith row of $\mathcal{T}(\omega)$. Thus, by exploiting sparsity in both the assumed input to the system and its measured output, the complexity of evaluating the spectral density is reduced to $O(N\,d)$, with N the number of points where the spectral density is being evaluated and d the dimension of the state-space. This implies significant computational savings compared to the worst case $O(N\,d^3)$ where all elements of $\mathbf{X}(t)$ and $\mathbf{P}(t)$ are needed. Such sparsity in input-output components is quite common in

scientific applications, since on one hand often experimentally only one or a small number of state variables of the system are accessible, and on the other hand experiments are often designed to disentangle inputs (e.g., a task provides highly specific excitation or observations are made when certain inputs are known to be dominant).

Other straightforward ways of improving the efficiency of such spectral density calculations are

- reducing the frequency resolution, e.g., by decimation or window-averaging;
- setting the spectral density to zero above a known frequency threshold of the system; and
- parallelizing the computation of the d^2 entries of $\mathcal{T}(\omega)$.

Use of the first two of techniques is problem-dependent, and care needs to be taken to ensure that the bias introduced is negligible. The results here can be easily extended to calculate cross-spectral densities of the multivariate process $\mathbf{X}(t)$. This is useful for problems where multiple components of the process are observed.

4.3.1 *Derivatives of the spectral density*

Some parameter estimation algorithms, such as Metropolis Adjusted Langevin Algorithm (MALA) require the likelihood to be differentiated. In order to apply such algorithms to likelihoods based on the spectral density, it is necessary to differentiate the spectral density. Derivatives can be approximated using finite differences. However, these can be time-consuming to compute, and inaccurate. Furthermore, the step-size in the finite difference approximation can only be decreased to a certain extent before rounding errors occur due to the limited machine precision. Here we derive analytic derivatives for the spectral density. As far as we are aware the derivation and use of analytic derivatives for spectral analysis of linear systems is novel. Some components of the derivation, for example eigenvalue and eigenvector derivatives, can be found in the literature on Automatic Differentiation, [Giles, 2008].

The results are only valid when all eigenvalues of the system are distinct. In the problems we have looked at this condition is satisfied for most of the parameter space, but not all of it. It is always possible to fall back on finite differences, if parameter sets with repeated eigenvalues are sampled.

For simplicity we consider the case of one time-varying input, and drop i and j subscripts on $\mathcal{T}_{ij}$. Note that the parameters themselves are real, not complex.

$$\frac{\partial}{\partial\theta}\left[f_X(\omega)\right] = \frac{\partial}{\partial\theta}\left[|\mathcal{T}(\omega)|^2\ f_P(\omega)\right] \tag{4.54}$$

$$= \left(2\,\mathrm{Re}[\mathcal{T}(\omega)]\mathrm{Re}\left[\frac{\partial\mathcal{T}(\omega)}{\partial\theta}\right] + 2\,\mathrm{Im}[\mathcal{T}(\omega)]\mathrm{Im}\left[\frac{\partial\mathcal{T}(\omega)}{\partial\theta}\right]\right) f_P(\omega) + |\mathcal{T}(\omega)|^2\ \frac{\partial f_P(\omega)}{\partial\theta} \tag{4.55}$$

To evaluate the derivatives of $\mathcal{T}(\omega)$, we need the derivatives of the eigenvalues and eigenvectors. Differentiating both sides of $A\mathbf{r} = \lambda\mathbf{r}$ gives us,

$$A_\theta\mathbf{r} + A\mathbf{r}_\theta = \lambda_\theta\mathbf{r} + \lambda\mathbf{r}_\theta, \tag{4.56}$$

where A_θ means each element of the matrix A is differentiated with respect to θ, and likewise for the right eigenvector $\mathbf{r}$ and $\mathbf{r}_\theta$. We require that right eigenvectors satisfy the constraint, $\mathbf{r}^T\mathbf{r} = 1$. Differentiating this constraint with respect to θ yields

$$\mathbf{r}^T\mathbf{r}_\theta = 0. \tag{4.57}$$

Combining Equations (4.56) and (4.57), and re-arranging, we obtain,

$$\left[\begin{array}{c|c} A - \lambda I & -\mathbf{r} \\ \hline \mathbf{r}^T & 0 \end{array}\right]\left[\begin{array}{c} \mathbf{r}_\theta \\ \hline \lambda_\theta \end{array}\right] = \left[\begin{array}{c} -A_\theta\ \mathbf{r} \\ \hline 0 \end{array}\right]. \tag{4.58}$$

There are a number of ways of deriving eigenvector derivatives. Imposing the constraint $\mathbf{r}^T\mathbf{r} = 1$ is somewhat arbitrary, and in particular is different from the standard constraint that is typically imposed on eigenvectors by numerical solvers, which is that the norm is unitary, i.e., $\mathbf{r}^*\mathbf{r} = 1$. Deriving eigenvector derivatives using this constraint is somewhat more complex - see Appendix A for details. Another derivation of eigenvector derivatives that imposes a somewhat different constraint can be found in [Giles, 2008].

The calculation above is a $(d+1) \times (d+1)$ linear system, in which we solve for $(\mathbf{u}_\theta, \lambda_\theta)$. We can implement a linear solver by performing a QR decomposition of the left-hand side matrix, which has computational complexity $\mathcal{O}(d^3)$. The remaining operations are matrix vector multiplications, which have computational complexity $\mathcal{O}(d^2)$. So, if we want to calculate derivatives with respect to p different parameters, the overall computational complexity is then $\mathcal{O}(d^3 + pd^2)$. The cubic complexity here is manageable in the regime we are interested in, i.e., when $p, d \approx 10 - 50$.

When p and d are both on the order of $10 - 100$ or greater, this means we get approximately a factor of d saving compared to the finite difference approach, which evaluates the eigenvalues and eigenvectors separately for each of the p parameters.

For the left eigenvectors we impose a different constraint: $\mathbf{l}^T\mathbf{r} = 1$, i.e., $\mathcal{LR} = \text{diag}[e_k] = I$. This means that parameter dependence is restricted to the R and L matrices in Equation (4.53). The linear system for obtaining the left eigenvector derivatives is then,

$$\left[\begin{array}{c|c} A^T - \lambda I & -\mathbf{l} \\ \hline \mathbf{r}^T & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{l}_\theta \\ \hline \lambda_\theta \end{array}\right] = \left[\begin{array}{c} -\mathbf{l}^T\ A_\theta \\ \hline -\mathbf{r}_\theta^T \cdot \mathbf{l} \end{array}\right]. \tag{4.59}$$

For second derivatives, we can simply differentiate Equations (4.56) and (4.57) again, to obtain another $(d+1) \times (d+1)$ linear system. This time the second derivatives of the eigenvectors and

eigenvalues are the unknowns. In this case the overall computational complexity for calculating the second derivatives with respect to p different parameters is $\mathcal{O}(d^3 + p^2 d^2)$, compared to $\mathcal{O}(p^2 d^3)$ for the finite difference calculation. Once the eigenvector and eigenvalue derivatives have been obtained it is straightforward to differentiate $\mathcal{T}(\omega) = \mathcal{R}\ \mathrm{diag}[1/(i\omega - \lambda_k)]\ \mathcal{L}$. In the worst-case scenario when all elements of $\mathcal{T}(\omega)$ are needed, this has a computational complexity of $\mathcal{O}(p\,n\,d^3)$ for first derivatives. In the best-case scenario where only one element is needed, it is $\mathcal{O}(p\,n\,d)$.

For the problems we are interested in $f_P(\omega)$ has an analytical form that can be differentiated by hand with respect to its parameters.

4.4 EVALUATING ACCURACY OF WHITTLE LIKELIHOOD

In Section 2.6 we discussed several different methods for evaluating the likelihood of a time-series conditional on model parameters. The Whittle likelihood can be orders of magnitude faster than particle filters and Kalman filters. However, if we want to use the Whittle likelihood to infer parameters that drive approximately linear dynamics we need to approximate the model by linearizing it around a stable fixed point and then further approximate the likelihood by neglecting covariance terms in the periodogram that only disappear in the limit as the length of the time series goes to infinity.

The purpose of this section is to explore the effect of these approximations on posterior inference by (i) comparing Kalman filters with a particle filter for a nonlinear model, (ii) comparing the Whittle likelihood with the Kalman filter for a linear model.

4.4.1 *Evaluating posterior accuracy under linearized model*

We first look at errors arising from linearizing the model. We are interested in cases when the dynamics of the nonlinear model are approximately linear around some stable equilibrium. The accuracy in the approximation depends on the magnitude of the input to the system, as this determines the amplitude of the system dynamics. For a given set of system parameters,

the linearization can be made arbitrarily accurate by choosing a small enough magnitude for the input.

However, this is not a strong enough result to guarantee accurate posterior inference. Suppose that the linearization is accurate for the true system parameters and the true input parameters. There may be regions of the parameter space where the likelihood is high under the linearized model but the linearization is inaccurate. For example, if a dynamical system has two stable fixed points and the magnitude of the noise that feeds into the system is sufficiently large, the system can transition between approximately linear dynamics around one fixed point to approximately linear dynamics around the other fixed point. In the linearized model, the system always stays relatively close to the stable equilibrium point that the system was linearized around.

It is not uncommon in dynamical systems models for the behaviour of the system to be different in different regions of the parameter space. Hence, for one parameter set the dynamics of the nonlinear model are approximately linear around a single fixed point, but for a different parameter set the system switches between two different fixed points. This is illustrated using a stochastic FitzHugh-Nagumo model in Figure 4.1.

In order to quantify the error caused by the linearization we ran both MCMC using a Kalman filter to estimate the likelihood (i.e. marginal MCMC), and particle MCMC, on data simulated from the FitzHugh-Nagumo model. We found the posterior distribution was noticeably broader in the marginal MCMC samples, i.e., in the algorithm where the model was linearized (results not shown). The computational cost of running particle MCMC on the FitzHugh Nagumo model was very high. It required more than one month of computing time on a cluster of 20 cores. This can be reduced, for example by using a low-level language, e.g., C, a more sophisticated variant of particle MCMC, e.g., a proposal that makes use of the next data-point, or by implementation on a computing environment where greater parallelism is possible, e.g., GPUs. All of these options are (currently) expensive in terms of user-time.

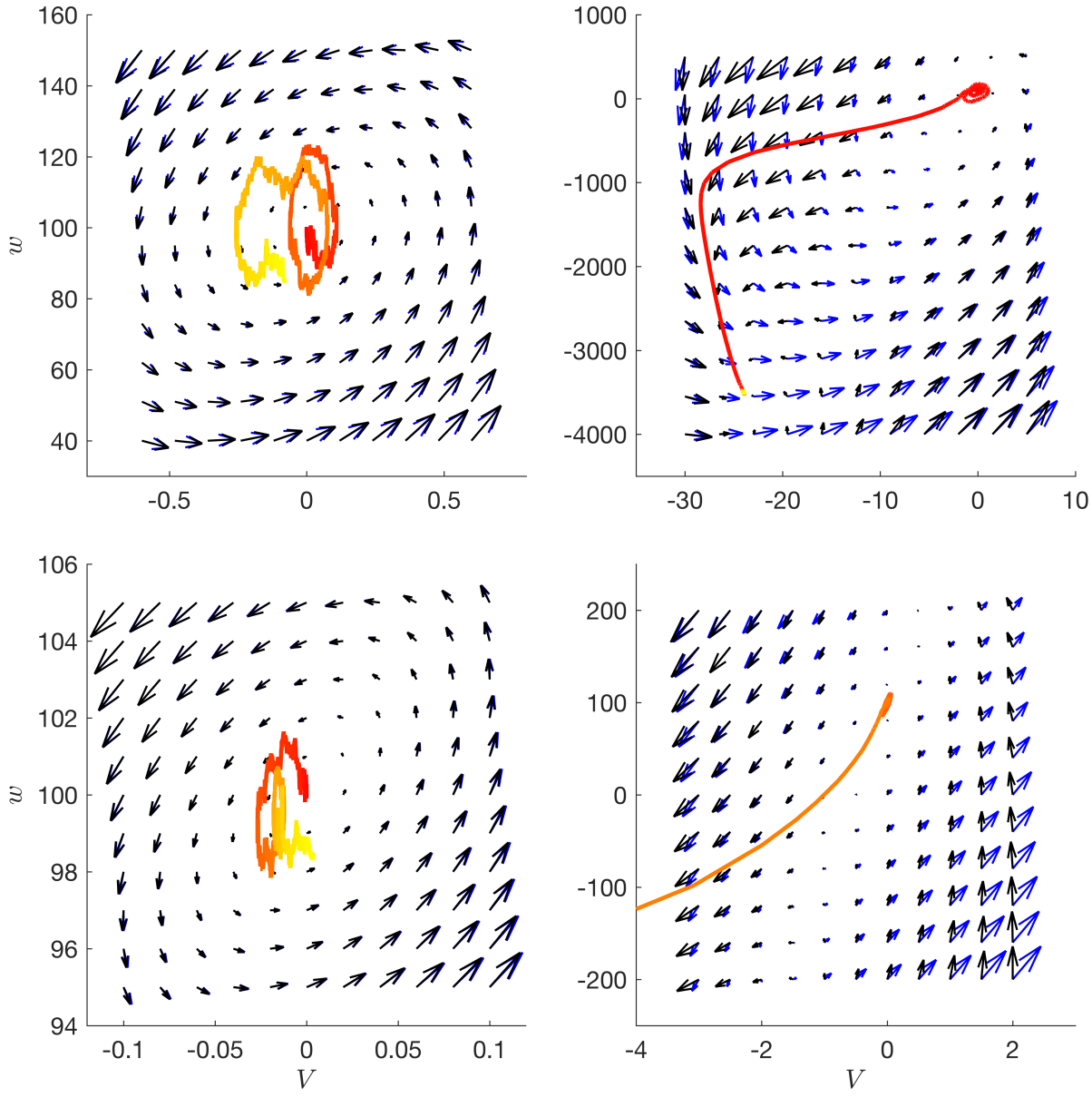

Figure 4.1.: Vector fields for the FitzHugh-Nagumo model (black arrows) and its linearization around a stable equilibrium (blue arrows), as well as sample path of nonlinear model (coloured line). Arrows represent the drift term, i.e., the deterministic time-derivative. *Left column:* Nonlinear model is well approximated by linearization. *Right column:* Linearized model is not a good approximation in spite of similar likelihood compared to the cases on the left, cf. see Figs. 4.2 and 4.3. $I(t) = 0$ and $P(t)$ was white noise with variance σ_{in}^2. *Top left* parameter values for $(a, b, c, d, I, \sigma_{in})$: $(-5, 6000, 40, 4000, 100, 100)$. *Top right*: $(-30, 6000, 40, 4000, 100, 100)$. *Bottom left*: $(-30, 6000, 40, 4000, 100, 10)$. *Bottom right*: $(-150, 3000, 170, 17000, 100, 10)$.

As an alternative to comparing with particle MCMC, the following checks are useful for partially quantifying the error that comes from linearizing the model:

i. Compare marginal likelihood estimates from the basic Kalman filter (i.e. linearizing about the stable point), Extended Kalman Filter (EKF), and particle filter on a 1-D grid for each individual parameter.

ii. Compare the posterior distribution obtained from running marginal MCMC with the Kalman filter and with the EKF.

iii. Simulate from the linearized model and the nonlinear model at parameter sets sampled by marginal MCMC to check for nonlinear dynamics.

These require much less computing time than running particle MCMC, and are also relatively easy to implement. We performed these checks for several scenarios with the aim of investigating the effect of our approximations.

In summary, when performing linearization around a fixed point we have observed the following useful properties:

i. There is only a small difference using the Kalman filter (in which the linearization is about a stable point) and the EKF when the input noise is small - Figure 4.2.

ii. As more data is collected, the posterior variance decreases at approximately the same rate with the approximate and exact methods - Figure 4.2.

iii. Our approximate posterior distribution tends to accurately capture the qualitative dependency structure between parameters - Figure 4.3.

iv. Where the likelihood differs between the Kalman filter and the EKF, we find that our approach tends to over-estimate uncertainty - Figure 4.2. This allows us to view sampled parameters as not ruled out yet (a concept that has proved useful for expensive climate

simulators [Williamson et al., 2013]) and also makes our approximation suitable for embedding within an exact approach to speed it up (e.g. using importance sampling or delayed acceptance [Golightly et al., 2015]).

Figure 4.2 shows that the error in the posterior is lower when the input noise and the observation noise are smaller. (We decrease both the noise parameters so that the signal to noise ratio does not change.) More specifically the Kalman filter and the EKF results are almost identical in the low noise setting, at least for the one-dimensional problem. The level of input noise controls the component of the error that comes from linearizing the model around a single point in the state-space, as opposed to linearizing locally around current state estimates (as in the EKF). The residual error, which exists when any form of linearization is done, is found by comparing the EKF with the particle filter. As is well known from previous studies, the EKF produces biased parameter estimates. Measured relative to posterior variance, the bias increases as more data is collected, see Figure 4.2. However, measured relative to the true parameter value, the bias in the posterior mean is small.

Figure 4.2 also shows that larger data-sets result in larger variances for particle filter estimates for a fixed number of particles, an issue that is discussed in [Kantas et al., 2015]. This reinforces the message in Section 2.6.4 that the number of particles needed to effectively apply particle MCMC makes the algorithm intractable for many problems of practical interest. Figure 4.3 shows that accuracy on one-dimensional problems does not necessarily imply accuracy in higher-dimensional problems with the same model. The true parameters in this case correspond to the bottom-left panel in Figure 4.2, i.e., the input noise is sufficiently low that the Kalman filter and EKF are almost identical in one dimension. For the higher dimensional problem (where a, b, c, I_0 are all unknown), we see that the Kalman filter over-estimates uncertainty, although it still captures the qualitative dependencies in the posterior.

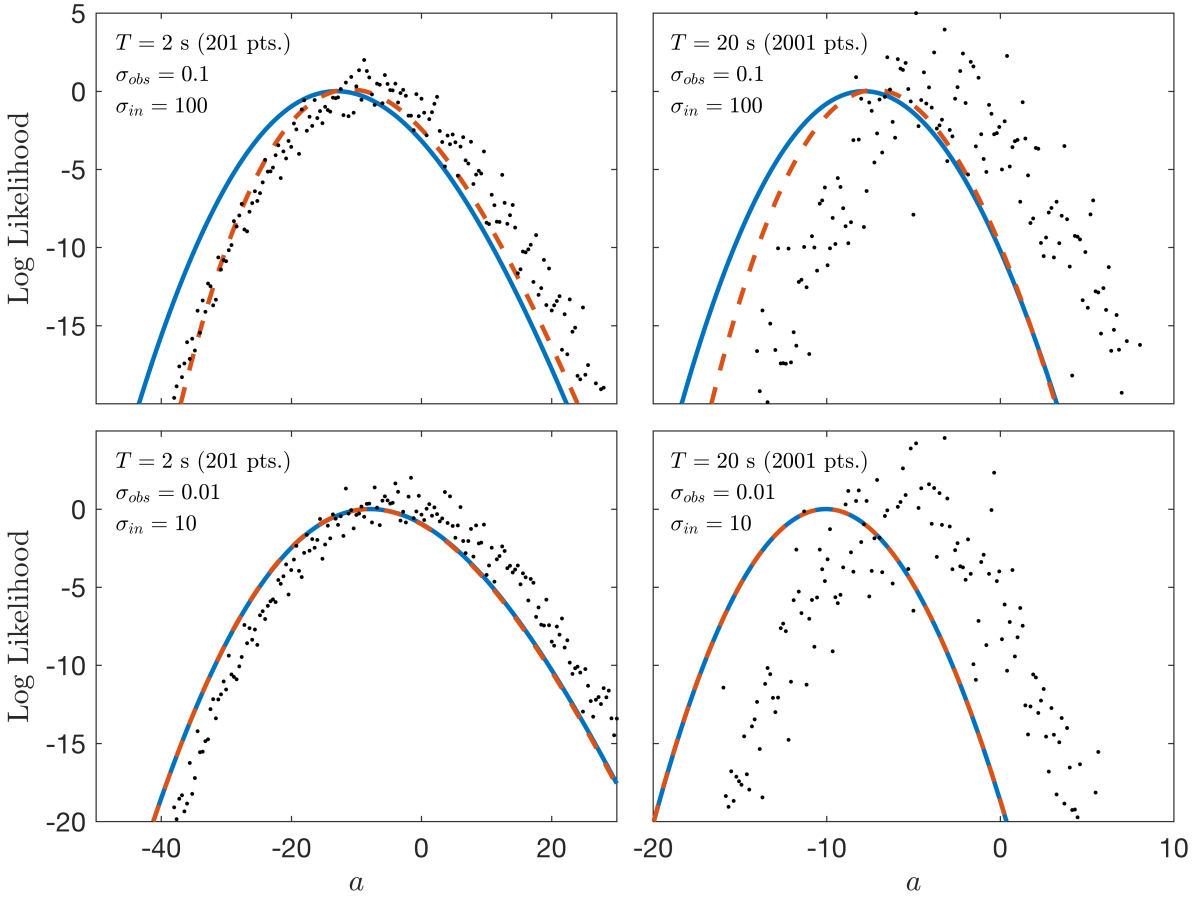

Figure 4.2.: Comparison between Kalman filter (solid blue line), EKF (dashed red line) and particle filter (black dots) for the stochastic FitzHugh-Nagumo model. $P(t)$ is white noise with variance σ_{in}^2, observation noise with variance σ_{obs}^2 is added. The only unknown parameter is a. The marginal likelihood is estimated on a uniformly spaced sequence of 200 a values. Fixed parameters: $b = 6000$, $c = 40$, $d = 4000$, $I_0 = 100$, $I(t) \equiv 0$. Time-step in solver and Kalman filter $= 10^{-3}$, in observations $= 10^{-2}$. Number of particles $= 1000$.

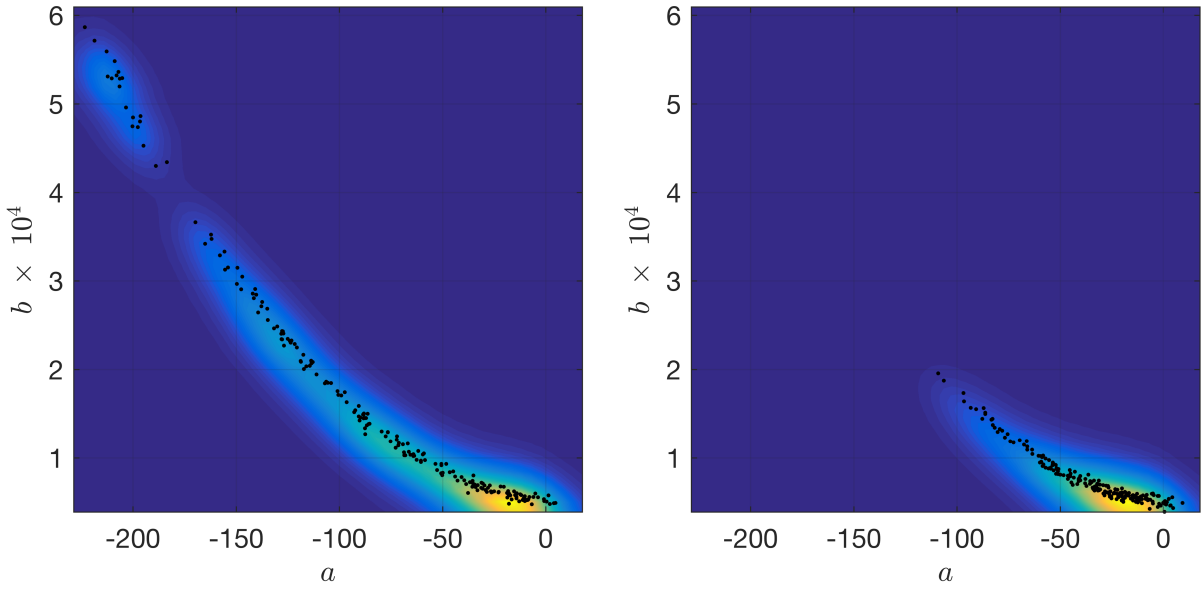

Figure 4.3.: Comparison between marginal MCMC with Kalman filter (left) and EKF (right) on a problem with unknown parameters (a, b, c, I_0). Plots show MCMC samples from the joint (a, b) marginal distribution only. Parameter values: $d = 4000$, $T = 2$, $\sigma_{in} = 10$, $\sigma_{obs} = 0.01$. Time-step in solver and Kalman filter $= 10^{-3}$, in observations $= 10^{-2}$. The MCMC chain started at the true parameters and ran for 10,000 iterations. Plots shows every 500th sample (black dots). The coloured background is a smoothed histogram with blue representing low probability density and yellow representing high probability density. The smoothing was done using the method in [Eilers and Goeman, 2004].

4.4.2 *Comparison between Kalman filter and Whittle likelihood on linear model*

The harmonic oscillator can also be written in the following form,

$$\frac{d^2}{dt^2}X(t) + 2\zeta\omega_0\frac{d}{dt}X(t) + \omega_0^2 X(t) = \epsilon(t) \tag{4.60}$$

The spectral density for the harmonic oscillator, driven by white noise, is a single peak if the oscillator is sufficiently under-damped $\zeta < 1/\sqrt{2}$. The location of the spectral peak is $\omega_r = \omega_0\sqrt{1-\zeta^2}$.

In Section 4.2.3, we saw that the accuracy of the Whittle likelihood depends on two quantities: ϕ given by Equation (4.42) and n, the length of the time series. Inspection of Equation (4.42) reveals that ϕ (and hence the error in the Whittle likelihood) is larger when the autocovariance is high at long lags, i.e., for strong correlations between time-points that are far away from each other in the discrete-time index. Further analysis has been done for the AR(1) process, [Contreras-Cristán et al., 2006], showing that a higher AR(1) coefficient (i.e., stronger correlation between time-points) lead to a larger error. For processes with a monotonically decaying autocovariance function it is clear that a slower decay in the autocovariance will mean that a longer time-series is needed to justify the use of the Whittle likelihood.

In the case of oscillatory processes, such as the linearized stochastic FHN equations and the Neural Population Models discussed in Chapter 3, the autocovariance function is oscillatory and decays to zero. There are then two distinct factors which affect the magnitude of ϕ. First is the frequency of oscillations, or equivalently the location of resonance peak(s) in the spectral density. If the systems oscillates at a lower frequency this results in an autocovariance function with a longer period. Second is the regularity of the oscillations, or equivalently the width of the resonance peak(s) in the spectral density. If the frequency of oscillations is highly consistent over time, as opposed to being more spread across frequencies, this results in an autocovariance function that decays more slowly.

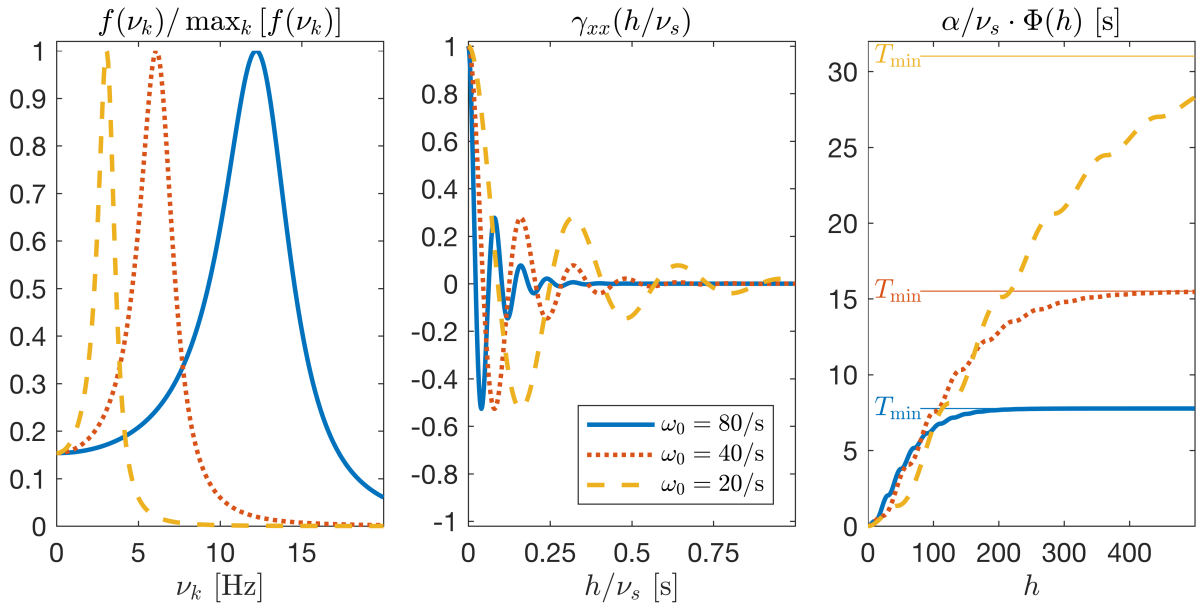

Figure 4.4.: Normed spectral density (*left*), autocovariance (*middle*), and accuracy heuristic (*right*) for the harmonic oscillator with damping $\zeta = 0.2$ and three different ω_0: 80/s (blue solid lines), 40/s (dotted red lines), and 20/s (golden dashed lines). Sampling frequency $\nu_s = 500$ Hz. The minimum time-series length can be calculated using Equation (4.61): $n_{\min} = \nu_s T_{\min} \equiv \alpha\phi = \alpha \lim_{h\to\infty} \Phi(h)$, where $\alpha \equiv (0.01 \cdot \max_k[f(\nu_k)])^{-1}$ and $\Phi(h) \equiv 2\sum_{j=0}^{h} |j||\gamma_{xx}(j/f_s)|$. $T_{\min}$ values for the different ω_0 are shown as thin lines in the *right* panel in matching colours.

Further quantification of the error can be done by comparing likelihoods from the Kalman filter (which is exact up to discretization error) with the Whittle likelihood. For our purposes, the Whittle likelihood is accurate when the posterior distribution under the Whittle likelihood is similar to the posterior distribution under the Kalman filter. By numerical experimentation, we have found that the Kalman filter and Whittle likelihood posteriors are similar for the damped harmonic oscillator when

$$\frac{\phi}{n} < 0.01 \max_k [f_Y(\omega_k)] \,. \tag{4.61}$$

As discussed above the Whittle likelihood is obtained by neglecting the ϵ_n terms in Equations (4.39)-(4.41). These ϵ_n terms were bounded by ϕ/n with $\phi = \sum_{m=-\infty}^{\infty} |m|\,|R(m \cdot \Delta t)|$ and n the length of the time-series. In order for the Whittle likelihood to be accurate we need the ϵ_n terms to be small relative to the spectral density, $f_Y(\omega)$. This motivates the threshold for ϕ/n in Equation (4.61).

The dependence of the error in the Whittle likelihood on ϕ and n is illustrated in Figure 4.5. In each subplot we see results for $T = 2$s and $T = 20$s, i.e. two different values of n. There are three pairs of plots (left, middle and right) which correspond to different parameter values for ω_0 (which are 80, 40, and 20). The spectral densities and the autocorrelation functions with each ω_0 value are plotted in Figure 4.4. We can see that decreasing ω_0 leads to slower oscillations (left panel), a more slowly decaying autocorrelation function (middle panel), and a higher minimum time-series length T_{min} in order for the Whittle likelihood to be accurate. For $\omega_0 = 80$ and $\omega_0 = 40$, the minimum time-series duration, T_{min}, is between 2s and 20s. In the left and middle panels of Figure 4.5 we see that the posteriors look different when $T = 2$s but almost identical when $T = 20$s. For $\omega_0 = 20$, the minimum time-series duration, T_{min}, is longer than 20s. Looking at the right panels of Figure 4.5 we see that the posteriors look different for both $T = 2$s and $T = 20$s. These numerical results support the use of Equation (4.61) as a heuristic rule to decide whether the error coming from the asymptotic approximation in the Whittle likelihood is sufficiently small to help justify its use. Computation of the quantity ϕ/n is relatively quick, so the heuristic is particularly useful for problems where using a Kalman filter to estimate the posterior distribution is prohibitively expensive, as is the case for the Liley *et al* model that we introduced in Chapter 3 and analyze in Chapters 5 and 7. As far as we are aware the development of a generally applicable heuristic, such as the one in Equation (4.61), is novel.

4.5 DISCUSSION

In this chapter we have shown how the asymptotic error that comes from working in the frequency domain can be quantified. And we have developed a heuristic that enables practitioners to determine how long a time-series needs to be in order for the Whittle likelihood to be accurate. We were also able to quantify the error in the marginal likelihood that comes from linearizing the model around a stable equilibrium by comparison with the Extended Kalman

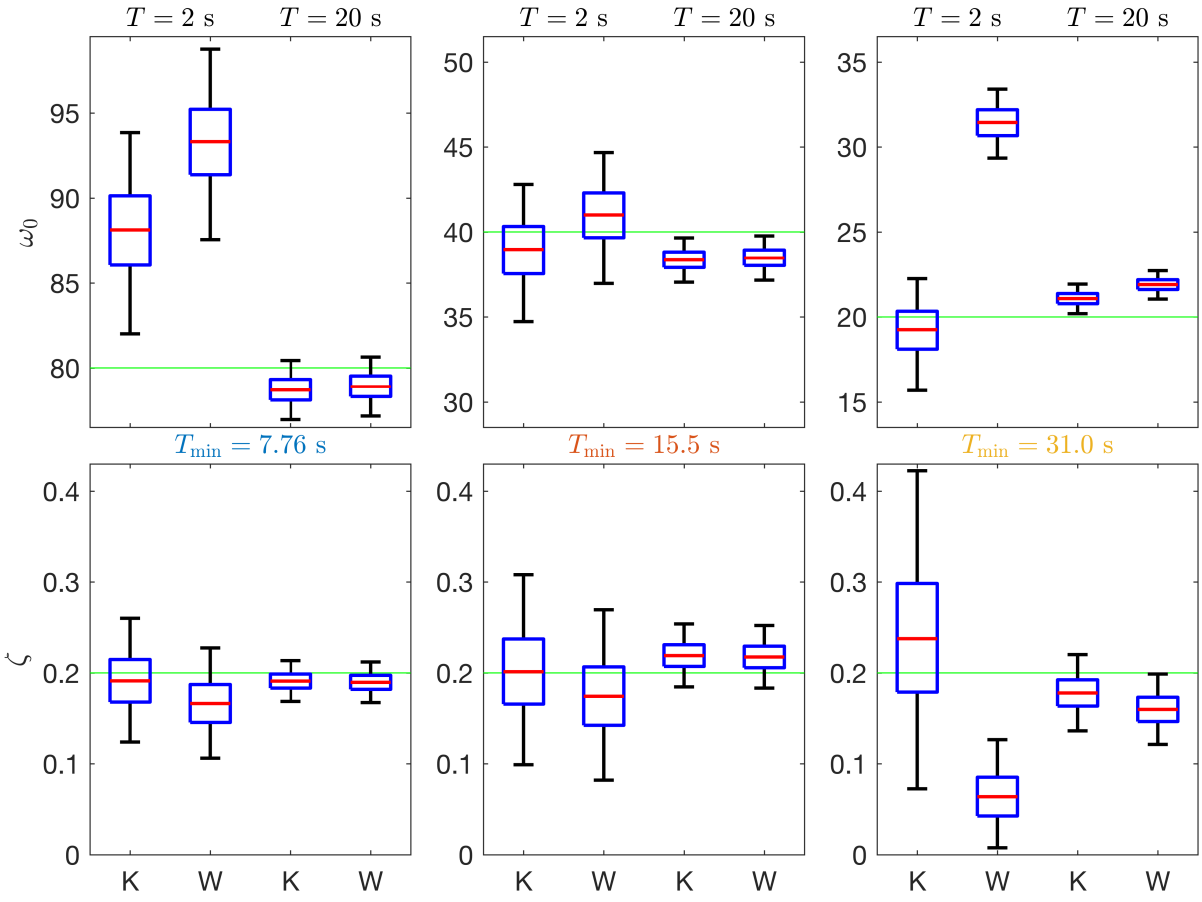

Figure 4.5.: Boxplots of the marginal posterior distributions for ω_0 (*top*) and ζ (*bottom*) under a uniform prior. The true parameters values are $\zeta = 0.2$ (all plots), and $\omega_0 = 80, 40, 20/s$ (from *left* to *right*). The exact Kalman filter (K) and the approximate Whittle likelihood (W) analyses are in agreement only if $T \geq T_{\min}$, as predicted by the heuristic, Equation (4.61), cf. Fig. 4.4 *right*.

Filter (EKF), which linearizes locally around the current state, and a particle filter, which gives unbiased estimates.

We have focused primarily on the case where the raw periodogram is used in the likelihood. However, it is not uncommon to use other estimates of the spectral density instead of the raw periodogram when evaluating the likelihood of model parameters using spectral analysis. It would be interesting to explore the accuracy of these likelihoods following a similar approach to the one developed in this chapter. The raw periodogram case was relatively simple because the sampling distribution of the raw periodogram is well characterized. In other cases, there may be more scope for approximating the sampling distribution by a Gaussian, but the error may be more challenging to quantify.

Our evaluation of the error coming from linearization is largely empirical, in that we compare posterior distributions under different levels of approximation in the likelihood. While it is clear how these approximations differ in the way they are constructed, there is little in the way of theoretical results for quantifying the error. This makes it difficult to generalize our results to other models, and leaves practitioners with a somewhat laborious task to thoroughly quantify the accuracy of posterior inference with a linearized model. While it would be interesting to explore the development of theoretical results for quantifying the error from linearization, this is a long-standing problem in the literature. It may be more fruitful to focus further research efforts on improving the efficiency of exact approaches for nonlinear dynamical systems.

5

EFFICIENT MCMC SAMPLERS FOR PARAMETER INFERENCE

As dynamical systems models increase in complexity, it becomes more challenging to sample efficiently from the posterior distribution using MCMC. In this chapter we introduce a novel type of reparameterization for differential equation models with dynamics around a stable equilibrium. We term these models stable differential equations. We show that the likelihood of these models is very sensitive to the value of the stable equilibrium. This leads us to parameterize the model in such a way that the equilibrium is treated as an unknown parameter of the model. This simplifies the geometry of the posterior distribution resulting in increased sampling efficiency, which we demonstrate on the Metropolis-within-Gibbs (MwG) and simplified manifold Metropolis Adjusted Langevin Algorithm (smMALA) MCMC samplers.

Improving scalability with both the dimension of the parameter space and the geometric complexity of the posterior distribution is increasingly relevant to a wide range of scientific applications. In many biological applications, domain scientists seek to develop detailed mechanistic models to explain emergent phenomena, leading to complex inference problems. We exemplify our approach on a simplified model that nevertheless illustrates the particular type of complexity we seek to address, and on a more complex Neural Population Model.

5.1 SENSITIVITY ANALYSIS OF LILEY *et al* MODEL

5.1.1 *Effect of changing a single parameter*

Figure 5.1 shows that changing a single parameter in the Liley *et al* model, in this case Γ_{ee}, has a noticeable effect on the spectral density of the model. The amplitude of the peak in the spectral density decreases and moves to a lower frequency. In Chapter 4 we saw that the spectral density of the model is determined by the Jacobian of the linearized model. The Jacobian is determined by the model parameters and, for the linearized model, it is evaluated at a stable fixed point, or steady-state, of the system.

The stable fixed point is a function of the model parameters. Changing one parameter in the model changes the fixed point of the system.

Figure 5.2 shows that changing multiple parameters in such a way that the steady state is kept the same has a smaller effect on the spectral density than changing only Γ_{ee}. Intuitively speaking, the model appears to be less sensitive to perturbations in directions that keep the steady-state fixed than to perturbations that change both a single parameter and the steady-state.

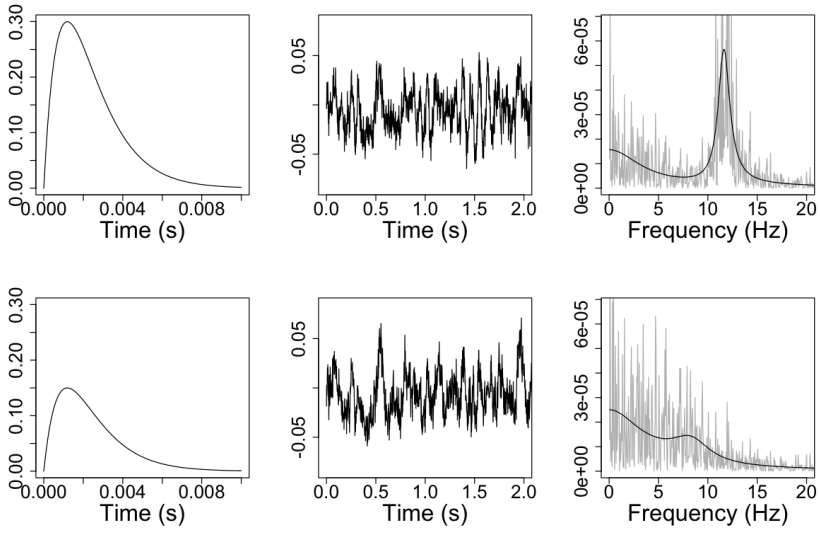

Figure 5.1.: Sensitivity of Liley *et al* model to single parameter change. *Left to right*: (i) synaptic response function, (ii) simulated time series, (iii) periodogram and spectral density.

	Γ_{ee}	h_e steady-state	Log Whittle Likelihood
Top Panels	0.3	-71.8	15,195
Bottom Panels	0.15	-73.6	15,005

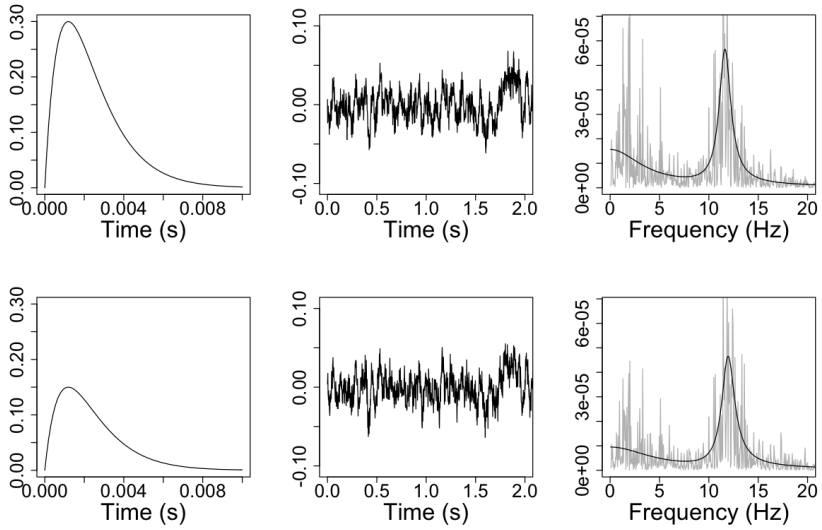

Figure 5.2.: Sensitivity of Liley *et al* model to multi-parameter change that keeps steady-state constant. *Left to right*: (i) synaptic response function, (ii) simulated time series, (iii) periodogram and spectral density.

	Γ_{ee}	h_e steady-state	Log Whittle Likelihood
Top Panels	0.3	-71.8	15,234
Bottom Panels	0.15	-71.8	15,217

5.1.2 *Grid-based likelihood evaluation*

Figure 5.3 consists of two likelihood surfaces for a simulated data-set. We refer to the parameters used to generate the simulated data as the true parameters, and the steady-state of the model given the true parameters as the true steady-state. It is relatively inexpensive to evaluate the likelihood on a grid because we have reduced the number of unknown parameters to two (γ_{ii} and Γ_{ee}). Under the assumption of uniform priors, the likelihood surface is equivalent to the posterior probability density. If we were to sample from this posterior density, we would find that γ_{ii} and Γ_{ee} would be correlated with each other. As can be seen in the top panel of Figure 5.3 the set of parameter values with steady-state equal to the true steady-state is correlated with the region of high probability density.

The bottom panel of Figure 5.3 is similar to the top panel, except that all points on the grid have the true steady-state. This is achieved using the method outlined in Section 5.2. In

contrast to the unconstrained likelihood surface, the parameters are now almost uncorrelated with each other in the posterior distribution.

If we were only interested in doing likelihood evaluations on a grid, none of this would matter a great deal. However, when it comes to sampling from the posterior distribution using an MCMC method (which for us is the only accurate and computationally feasible method when there are more than 3-4 unknown parameters), we need to be able to design parameter updates that move around the parameter space efficiently. As we will see in the next section, applying the steady-state constraints improves the efficiency of MCMC by reducing the geometric complexity of the posterior distribution.

5.2 MODEL REPARAMETERIZATION: EQUILIBRIUM VALUES AS PARAMETERS

A dynamical system is specified by supplying a set of parameters θ. Hence, generally its equilibria $\mathbf{x}^*$ will be a function of these θ. As discussed in Section 4.1.3, a system is at equilibrium if the derivatives of all the state variables are zero. Equilibria can be stable, in which case the system returns to its equilibrium following a small perturbation, or unstable, in which case a small perturbation causes the system to diverge away from the equilibrium. In the context of SDEs perturbations come in the form of noise, and the methods we develop in this chapter are targeted towards models where the noise is sufficiently small that the system dynamics operate in a small neighbourhood of a stable equilibrium, models which we refer to as stable SDEs. The methods are also applicable to stable ODEs where perturbations come in the form of transient deterministic input, and where the system returns to the stable equilibrium following this input.

For the form in Eq. (4.17) discussed in the previous chapter, equilibria are identified by solving $\mathbf{F}[\mathbf{x}^*;\theta] = 0$. For stable ODEs, $\mathbf{x}^*$ does not explicitly enter the likelihood calculations. Nevertheless, the likelihood is implicitly a function of $\mathbf{x}^*$, and the likelihood can be more sensitive to values of $\mathbf{x}^*$ than to values of θ. For stable SDEs, the likelihood calculations we use depend on linearizing the system around $\mathbf{x}^*$. In this case the likelihood is explicitly a function of $\mathbf{x}^*$.

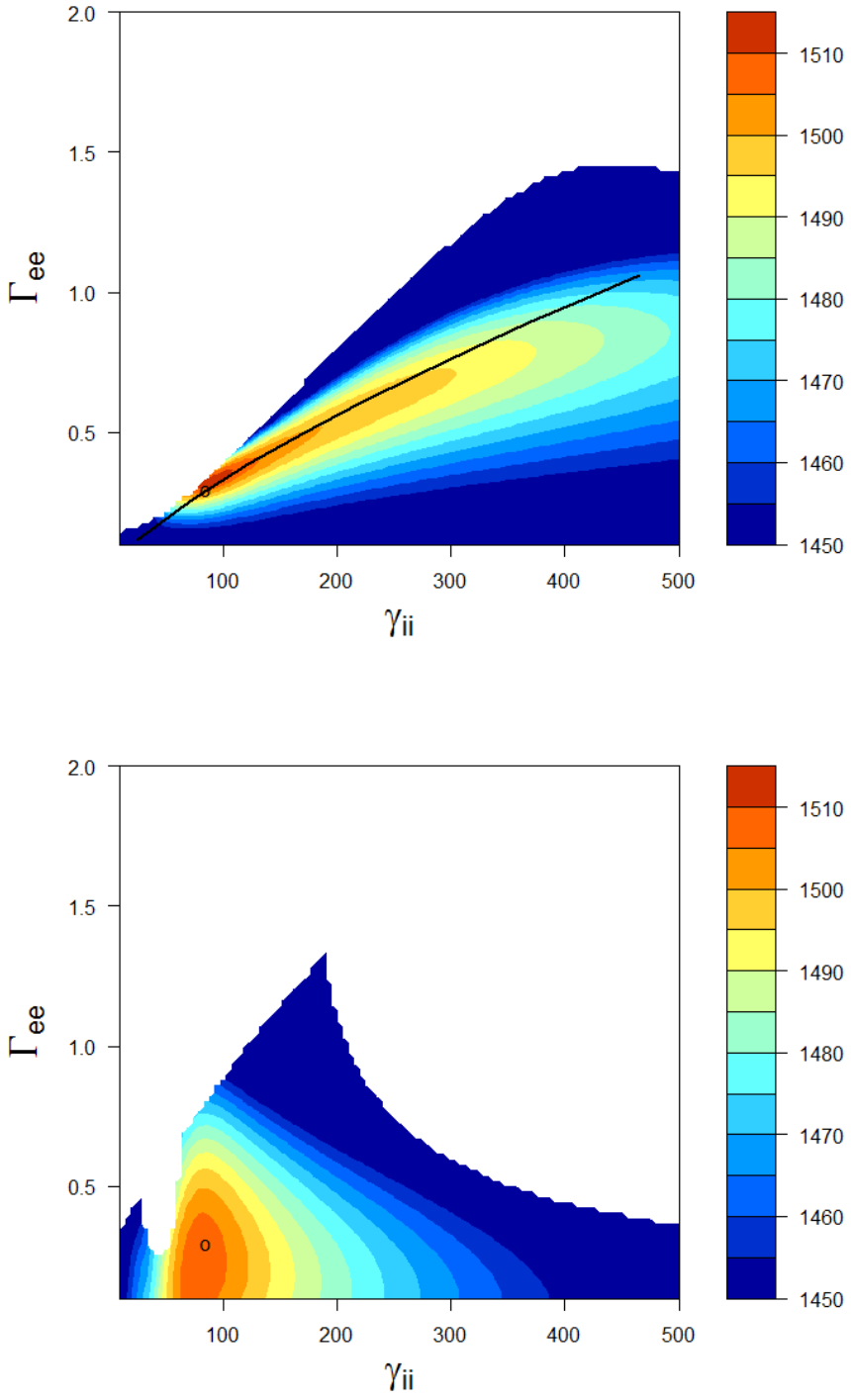

Figure 5.3.: The log-likelihood of a simulated data-set evaluated on a 2-D grid. *Top:* without steady-state constraints. *Bottom:* with steady-state constraints.
In the white region, the fixed point of the model is unstable. The circle is the location of the true parameter set. The line in the top plot is the set of parameters that have the true steady-state.

In this chapter we use an MCMC algorithm to explore the posterior on θ, and find that the posterior may be explored more efficiently by reparameterizing the model such that $\mathbf{x}^*$ is treated directly as a parameter of the model. Thus rather than treating $\mathbf{x}^*$ as a function of θ (as in the original parameterization), some elements of θ can be calculated as a function of $\mathbf{x}^*$. If θ is of dimension d_θ and $\mathbf{x}^*$ of dimension d_x, then the combined parameter space $(\theta, \mathbf{x}^*)$ has dimension $d_\theta + d_x$ but only d_θ degrees of freedom. One could choose any combination of d_θ parameters to parameterize the model. Then the remaining d_x parameters would be a function of the chosen d_θ parameters. We therefore introduce the subscripts s and d to distinguish between these parameter subsets: the MCMC is applied to the reparameterized space θ_s, with θ_d being a deterministic function of θ_s and $\mathbf{x}^*$.

As an example, suppose that the equations for the equilibrium points were of the simple form

$$\mathbf{F}[\mathbf{x}^*, \theta_s] + \theta_d = 0, \tag{5.1}$$

where $\mathbf{F}$ is a nonlinear function with d_x components, θ_d is a vector of parameters also with d_x elements, and $\theta = (\theta_s, \theta_d)$. Then we can simply reparameterize the model in terms of $(\theta_s, \mathbf{x}^*)$, and calculate $\theta_d = -\mathbf{F}[\mathbf{x}^*_s, \theta_s]$. Let us compare the form in Eq. (5.1) with the one that arises for our systems of interest from Eq. (4.17). Note that above we had defined $\mathbf{p}(t;\theta) \equiv 0$ as default for the input. What if instead either stimuli or resting state fluctuations occurred on top of some constant background input $\overline{\mathbf{p}} + \mathbf{p}(t;\theta)$? We can bring this into our standard form by absorbing the constant background input $\tilde{\mathbf{F}}[\mathbf{x}, \theta] \equiv \mathbf{F}[\mathbf{x}, \theta] + \overline{\mathbf{p}}$, and using $\tilde{\mathbf{F}}$ instead in Eq. (4.17). Then solving for equilibria of $\tilde{\mathbf{F}}[\mathbf{x};\theta] = 0$ naturally has the form of Eqn. (5.1), with the background input $\overline{\mathbf{p}}$ serving as the θ_d.

If the dimensions of $\mathbf{F}$ and θ_d do not coincide, i.e., if in our application a background input is added only to some equations of the system, then the equilibrium calculation can be split as follows

$$\mathbf{F}_1[\mathbf{x}^*, \theta_s] = 0, \tag{5.2}$$

$$\mathbf{F}_2[\mathbf{x}^*, \theta_s] + \theta_d = 0, \tag{5.3}$$

where $\mathbf{F} = (\mathbf{F}_1, \mathbf{F}_2)$, and now $\mathbf{F}_2$ and θ_d have the same dimension, i.e., in our application only these equation receive background inputs. The previous Eq. (5.1) then is the special case of $\mathbf{F}_1 = \emptyset$. Now the model can be reparameterized in terms of $(\theta_s, \mathbf{x}_2^*)$, where $\mathbf{x}^* = (\mathbf{x}_1^*, \mathbf{x}_2^*)$ and $\mathbf{x}_2^*$ has the same dimension as θ_d. Please note that while $\mathbf{F}_1$ and $\mathbf{x}_1^*$ have the same number of dimensions, as do $\mathbf{F}_2$ and $\mathbf{x}_2^*$, the partitioning of state components can differ between $\mathbf{F}$ and $\mathbf{x}^*$, since we can freely choose which components of $\mathbf{x}^*$ to trade for the θ_d. We can write our solutions as

$$\mathbf{x}_1^* = \mathbf{G}_1[\theta, \mathbf{x}_2^*], \tag{5.4}$$

$$\theta_d = \mathbf{G}_2[\theta_s, \mathbf{x}^*] = \mathbf{G}_2[\theta_s, (\mathbf{G}_1[\theta_s, \mathbf{x}_2^*], \mathbf{x}_2^*)] \equiv \tilde{\mathbf{G}}_2[\theta_s, \mathbf{x}_2^*], \tag{5.5}$$

Note that we assume only that a solution can be obtained numerically, where $\mathbf{G}_1$ is the solution corresponding to $\mathbf{F}_1$, and $\mathbf{G}_2$ to $\mathbf{F}_2$, respectively. The number of unknowns and the number of constraining equations is the same as in the original equations for the steady-state: $\dim(\mathbf{x}_1^*) + \dim(\theta_d) = \dim(\mathbf{x}^*) = \dim(\mathbf{F})$.

If the solutions in Equations (5.4)-(5.5) can be obtained in closed form by algebraically rearranging the equations, then this obviously has practical advantages for the following computations. This will be the case for the examples we consider in the following. Thus given $(\theta_s, \mathbf{x}_2^*)$ through an MCMC update, we can compute $(\mathbf{x}_1^*, \theta_d)$ directly by substitution into known formulae.

Prior distributions may be specified on the original parameter space. In this case, if we are proposing parameters on the reparameterized space, we need to evaluate a Jacobian determinant in order to correctly evaluate the prior density in the MCMC acceptance ratio,

$$p(\theta_s, \mathbf{x}_2^*) = p(\theta_s, \theta_d)\ |\mathcal{J}_{\mathbf{T}}|, \text{ where } \mathbf{T} : (\theta_s, \mathbf{x}_2^*) \to (\theta_s, \theta_d) = (\theta_s, \tilde{\mathbf{G}}_2[\theta_s, \mathbf{x}_2^*]). \tag{5.6}$$

The Jacobian determinant, $|\mathcal{J}_{\mathbf{T}}|$, can be evaluated analytically if $\tilde{\mathbf{G}}_2$ is available in closed form. If not, $|\mathcal{J}_{\mathbf{T}}|$ can be approximated numerically using finite differences (yielding a noisy MCMC algorithm [Alquier et al., 2016], not considered further in this paper).

When and why might this reparameterization be effective? The answer to this depends on the structure of the target distribution, and the algorithm used to sample it. The target distribution may have no dependencies, or only very weak dependencies, between model parameters. The reparameterization is not a useful tool in that setting. Target distributions with likelihoods derived from differential equation models are typically more complicated, with often strong dependencies – linear and/or nonlinear – between model parameters. The reparameterization simplifies the posterior so that fewer MCMC iterations are required, and in fact for the model analyzed in Sections 5.3.3-5.3.4, also the computational cost per MCMC iteration is reduced by the reparameterization, because it is no longer necessary to numerically solve a nonlinear equation for the steady-state values.

Consider sampling from a complex target distribution with an MCMC algorithm where the proposal does not take account of the covariance structure in the target distribution (such as the Metropolis-within-Gibbs algorithm in Section 2.3.2). As demonstrated in Section 5.1, the likelihood is often very sensitive to the location of the steady-state, and steady-state is a complex function of the model parameters. This leads to strong correlations in the likelihood. If the dependencies between elements of $(\theta_s, \mathbf{x}_2^*)$ are weaker than the dependencies between elements of θ, then it will be easier to sample on the reparameterized space than on the original space.

The reparameterization method can also be useful in MCMC algorithms that do take account of the covariance structure in the target distribution (such as the smMALA algorithm in Section 2.3.4). If the curvature of the target distribution is constant, we would not expect the reparameterization method to be any more efficient in terms of the Effective Sample Size (ESS). However, if the equilibrium point is a nonlinear function of the model parameters, then this may cause the target distribution to have non-constant curvature. In the reparameterization, θ_d will be a nonlinear function of $(\theta_s, \mathbf{x}_2^*)$. If the likelihood is more sensitive to $\mathbf{x}_2^*$ than it is to θ_d, then we would expect sampling on the reparameterized space to be more efficient than on the original space. In this situation, the gain in efficiency is obtained by reducing nonlinearity in the likelihood function.

Examples of applying the reparameterization method, along with further specific explorations of its effectiveness, can be found in Sections 5.3. However, we want to draw attention to two further issues that are relatively common in practice. First, it is possible that the map $\theta \to \mathbf{x}^*$ is not well-defined because there are multiple stable equilibria for one given set of parameters θ. In this case the following map is still well-defined for stable ODEs: $(\theta, \mathbf{x}_0) \to \mathbf{x}^*$, where $\mathbf{x}_0$ is the initial condition of the system. For SDEs with multiple stable equilibria, there is always a positive probability that the system will jump between the basins of attraction of the different stable states. The methods presented in this paper are theoretically not applicable in this setting. However, in practice the system may remain close to one specific stable state during the entire interval when the observations, y, are collected. If so, then it is possible to construct a map $(\theta, y) \to \mathbf{x}^*$ that is de facto well defined. The likelihood for this de facto mono-stability typically increases with a decrease of the strength of the noise driving the system.

Second, the proposal densities $q[(\theta_s, \theta_d) \to (\mathbf{x}_1^*, \mathbf{x}_2^*)]$ and $\tilde{q}[(\theta_s, \mathbf{x}_2^*) \to (\mathbf{x}_1^*, \theta_d)]$ could have support on different regions of $\mathcal{R}^{d_\theta + d_x}$. For example, prior knowledge may dictate that for $\theta_d < 0$ one has $q = 0$ but this is not explicit in the reparameterized $\tilde{q} \neq 0$. If there is such strong prior information then more work needs to be done to ensure that q and $\tilde{q}$ have the

same support. For example, it may be that $\theta_d > 0$ implies $\mathbf{x}_2^* > \mathbf{x}_0$. If so, then an appropriate $\tilde{q}[(\theta_s, \mathbf{x}_2^*) \to (\mathbf{x}_1^*, \theta_d)]$ would be a normal distribution on $\log[\mathbf{x}_2^* - \mathbf{x}_0]$.

5.3 EVALUATING EFFICIENCY OF REPARAMETERIZATION

5.3.1 *MwG on deterministic FitzHugh Nagumo equations*

Using the FitzHugh Nagumo equations as presented in Equations (4.20)-(4.21), we set $P(t) = 0$ throughout, but $I(t) = 100$ for the time interval $1 < t < 1.1$ and $I(t) = 0$ otherwise. The initial conditions are set to the stable equilibrium. This is a function of the model parameters, so the initial conditions change when the parameters are updated. It would not be difficult to extend the methods here to the case when the model parameters and the initial conditions are treated as separate parameters of the inference problem. The numerical solution and parameter values used are shown in Fig. 5.4, along with numerical solutions at two other parameter sets.

We make the following assumptions about the inference problem:

- We can only observe V with observation noise, and do not have any observations for w.
- The value of d is known.
- The prior distributions for a, b, c, I_0 are uniform over a large (unspecified) range.

Scatterplots for selected pairs of parameters (obtained using the reparameterized algorithm) are shown in Figure 5.5.

Figure 5.4 illustrates how the likelihood of the data is sensitive to the steady-state value V^*. This induces strong correlations in the posterior distribution between the original model parameters, c and I_0, seen in Figure 5.5. Reparameterizing the likelihood using the steady-state value results in a likelihood function with weaker correlations (Figure 5.5). This means that that the MwG algorithm should be able to sample the parameter space more efficiently. This is evaluated empirically below.

A Metropolis-within-Gibbs MCMC algorithm (Algorithm 3 in Chapter 2) was run with and without model reparameterization. In both cases, the parameters are updated using a Gaussian

distribution centred on the current parameter value. The proposal width was tuned manually using preliminary runs with a procedure that estimates the conditional standard deviation in the posterior distribution [Raferty and Lewis, 1996]. This resulted in different proposal widths between the two parameterizations.

We evaluated the performance of each MCMC algorithm by estimating the Effective Sample Size (ESS) for each parameter using the LaplacesDemon package in R. There is a wide range in the ESS values across parameters. This is because some parameters are largely independent of the other parameters (leading to a high ESS). However, there are also strongly correlated parameter pairs (leading to a low ESS). When the model is reparameterized the minimum ESS is around 14 times higher than it is with the original parameterization; the same is true of the average ESS (see Table 5.1).

It is worth noting that the difference in the ESS is not the same for all parameter sets. Indeed there are some scenarios where the ESS decreases for some of the parameters when the model is reparameterized (not shown). It is also worth noting that the ESS in the reparameterized model is still only a small fraction of the number of MCMC iterations, around $0.1 - 0.5\%$. These observations suggest that even when the model is reparameterized, the MCMC mixes relatively slowly. Depending on the model and computational resources available this may or may not be a problem. If it is problematic, as we have found to be the case for the Neural Population Model considered later, then algorithms based on Langevin or Hamiltonian dynamics may perform better. If the mixing rate is not problematic, MwG with reparameterization offers an easy to implement alternative to Langevin and Hamiltonian Monte Carlo methods. Since MwG does not require costly derivative calculations, there may also be situations when the overall computational efficiency is higher (as measured by ESS / CPU-time).

5.3.2 *Justifying the use of Whittle likelihood for EEG analysis*

We are interested in the Neuroimaging problem of inferring NPM model parameters from EEG data. The characteristics of EEG datasets vary depending on the experimental setup and exper-

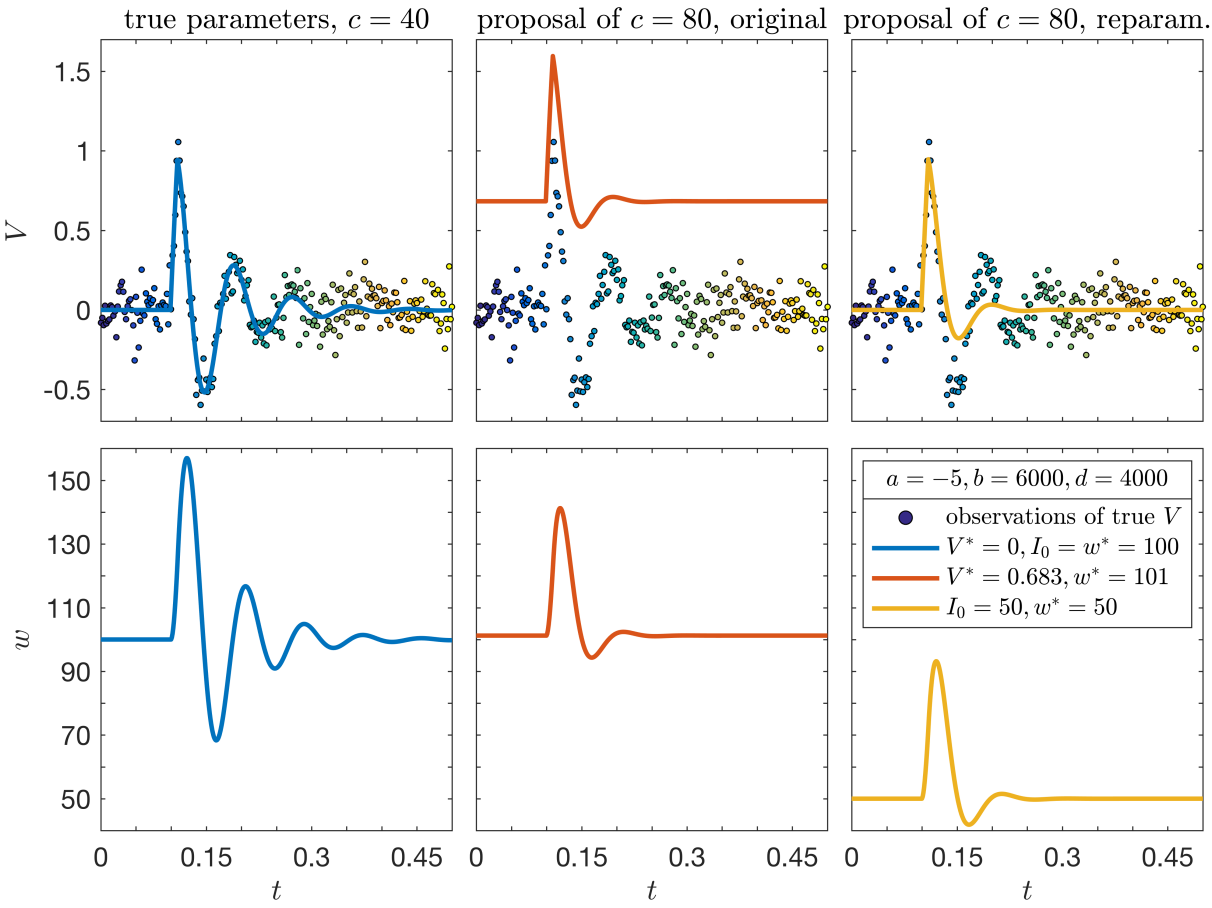

Figure 5.4.: Illustrative reparameterization example. *Left column*: Solutions (blue solid lines) for V (top) and w (bottom) at true parameter values: $a = -5$, $b = 6000$, $d = 100$, as well as $c = 40$ and $I_0 = 100$. The same noisy observations of the true V solution (dots, colour changing with t) are shown here and in the other columns. *Middle column*: A proposal of $c = 80$, with all other parameters remaining at true values, changes (V^*, w^*) in the original scheme. The proposed solution (orange solid line) deviates strongly from the observations of V. *Right column*: The same proposal in the reparameterized scheme changes (I_0, w^*). Since $V^* = 0$ remains unchanged, deviations from the observations of V remain limited.

	Original parameterization		Steady-state reparameterization	
Parameter	Proposal s.d.	ESS	Proposal s.d.	ESS
a	5	13	3.5	175
b	250	22	250	144
c	1	10	3	174
I_0	2.5	9	n/a	249
	Average ESS	**13.5**	Average ESS	**185.5**

Table 5.1.: Results for Metropolis-within-Gibbs MCMC algorithm with and without reparameterization. Total number of MCMC iterations was 100,000. ESS was calculated on the second half of the samples (i.e. 50,000 samples) to remove the effects of burn-in. Proposal standard deviation (s.d.) for V^* in the reparameterized algorithm = 0.015.

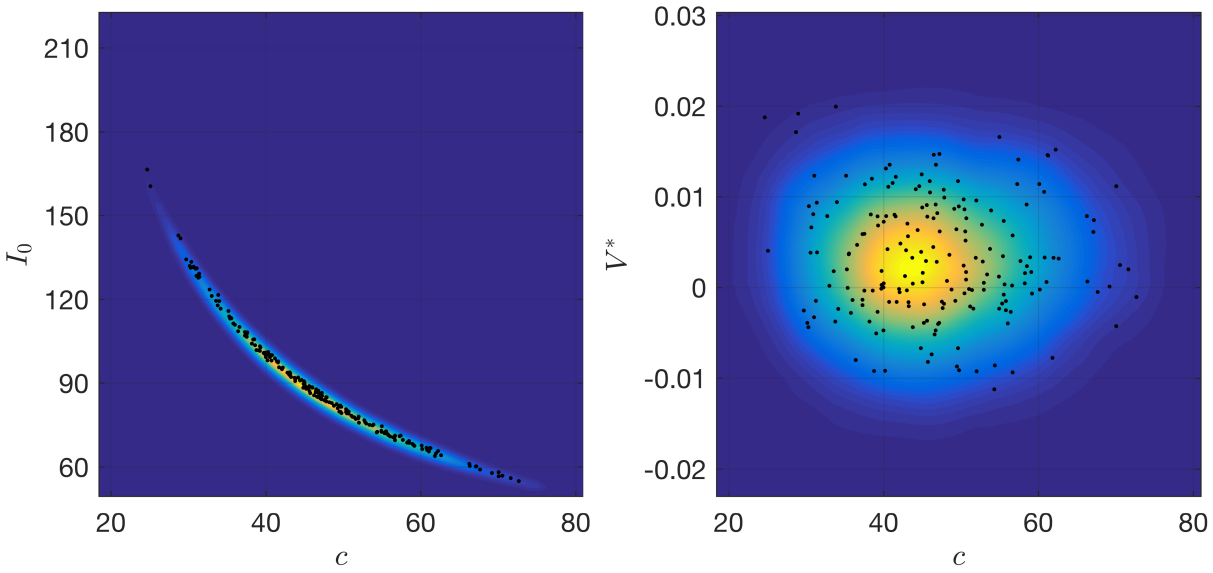

Figure 5.5.: MCMC results for deterministic FHN. The posterior correlation between c and V^* (*right panel*, reparameterized) is much weaker than the correlation between c and I_0 (*left panel*, original). The MCMC chain was started at the true parameter set ($a = -5$, $b = 6000$, $c = 40$, $I_0 = 100$) and run for 100,000 iterations. Every 500th sample (black dots) and a smoothed histogram (background colour map) are shown, as in Figure 4.3. Quantiles (0.025, 0.5, 0.975) of posteriors for parameters not shown were a: (-11, 4.11, 15.4) and b: (4980, 5400, 6250).

imental subjects being studied. Event Related Potentials (ERPs) and epileptic seizure dynamics typically feature non-stationary and/or nonlinear behaviour [David et al., 2005, Andrzejak et al., 2001].

In contrast, adult resting-state data is typically stationary with a power spectrum that is a superposition of an inverse frequency ($1/f$) curve with a peak in the range $8 - 12$Hz. It has been observed, [Bojak and Liley, 2005], that the location and amplitude of the peak in the power spectrum can change with the anaesthetic drug concentration, something that we return to in Chapter 7. Here we restrict ourselves to inferring the parameters when the true parameter values are static.

The Liley *et al* NPM introduced in Chapter 3 has several different dynamical regimes, e.g. linear around a stable fixed point, oscillatory, or chaotic. Our working assumption is that if the data is stationary, the underlying process can be modelled by an NPM with parameter values that lead to approximately linear dynamics around a stable fixed point.

See Section 4.4 for a detailed discussion of the effect of linearizing the model. For the FitzHugh Nagumo model we found that using the marginal likelihood calculated with the linearized model resulted in a posterior distribution that over-estimated uncertainty. The same may be true here although it is more difficult to test because of the greater computational resources required to run the Kalman filter, Extended Kalman filter and particle filter. We have simulated from the model at parameter sets sampled by the MCMC algorithm and found a good agreement between nonlinear and linearized model behaviour, consistent with the results in [Bojak and Liley, 2005].

For this problem, particle MCMC is not computationally tractable, so we cannot make a direct comparison of the different likelihood choices.

Below we present MCMC results for parameter estimation in the Neural Population Model with approximately linear dynamics. We use the Whittle likelihood and the smMALA algorithm (Algorithm 4 in Chapter 2).

For this problem the Whittle likelihood is over 100 times faster than the Kalman filter. This is because there is only one non-zero input, and we only observe one component, so the likelihood calculation is faster by a factor of $O(d^2)$. For this model $d = 14$.

Following the discussion in Section 4.4.2, we use the definition of ϕ in Eq. (4.42) and the heuristic in Eq. (4.61) to determine the minimum time series length for which we expect the Whittle likelihood to be accurate. For this model, the spectral density is calculated using the method in Section 4.3 and the autocovariance function is calculated by numerically approximating,

$$\gamma_{xx}(h) = \int_{-1/2}^{1/2} \exp[2\pi i\, \omega h]\, f(\omega)\; d\omega \tag{5.7}$$

Assuming the model and problem parameters in Tables 5.3, the result of this analysis is a minimum time series length of around $7,000$ time-points, which is equivalent to around 14s of data.

5.3.3 *MwG on stochastic Liley et al model*

In this section we apply the MwG algorithm to the Liley *et al* model. In contrast to the previous section the model is stochastic and we evaluate the Whittle likelihood, introduced in Section 4.2.3, at each sampled parameter set. We assume that 8 of the parameters are unknown, and that the remaining parameters are known, i.e., have fixed values, including the steady-state of the system. We found that when we increased the number of unknown parameters beyond 8, the MwG algorithm failed to converge. For this model we found that the smMALA algorithm scales better with the number of unknown parameters, see Section 5.3.4 for results.

5.3.3.1 *Reparameterizing the Liley et al model*

For the Liley *et al* NPM defined by Equations (3.28)-(3.29), the equations of the form (5.4) can be obtained by re-arranging the steady-state equations for short-range and long-range connectivity,

$$I^*_{ee} = q_{ee}\left[N^{\beta}_{ee}S_e(h^*_e) + \Phi^*_{ee} + \bar{p}_{ee}\right], \tag{5.8}$$

$$I^*_{ei} = q_{ei}\left[N^{\beta}_{ei}S_e(h^*_e) + \Phi^*_{ei} + p_{ei}\right], \tag{5.9}$$

$$I^*_{ik} = q_{ik}\left[N^{\beta}_{ik}S_i(h_i)\right], \tag{5.10}$$

$$\Phi^*_{ee} = N^{\alpha}_{ee}S_e(h^*_e), \tag{5.11}$$

$$\Phi^*_{ei} = N^{\alpha}_{ei}S_e(h^*_e). \tag{5.12}$$

The remaining steady-state equations are,

$$0 = h^r_e - h^*_e + \psi_{ee}(h^*_e)I^*_{ee} + \psi_{ie}(h^*_e)I^*_{ie}, \tag{5.13}$$

$$0 = h^r_i - h^*_i + \psi_{ei}(h^*_i)I^*_{ei} + \psi_{ii}(h^*_i)I^*_{ii}. \tag{5.14}$$

Equations $(5.8) - (5.12)$ can be used to eliminate I^*_{ee}, I^*_{ei}, I^*_{ie}, I^*_{ii} from (5.13)-(5.14). The resulting equations are of the form (5.5) with $\mathbf{x}^*_2 = (h^*_e, h^*_i)$, and $\theta_d = (\bar{p}_{ee}, p_{ei})$. Note that the solutions for θ_d are easily obtained in closed form.

5.3.3.2 *MCMC diagnostics*

Each iteration of an MCMC algorithm involves proposing a new parameter set and then either accepting or rejecting that move. In this section we use the acceptance rate, i.e., the frequency with which proposals are accepted, as a diagnostic for the sampling efficiency of our MCMC algorithms. A higher acceptance rate is an indication that the MCMC algorithm is exploring the posterior distribution more efficiently. For more detail see Section 2.3.

Table 5.2 shows that the acceptance rates of the MCMC algorithm increase when the steady-state constraints are applied. Also, the burn-in of the algorithm, the number of MCMC iterations required to reach regions of high probability density, decreased from around $25,000$ for the no constraints algorithm to around $1,000$ with the steady-state constraints.

Table 5.2.: Acceptance rates for MCMC

Parameter	No constraints	Steady-state constraints
Γ_{ee}	3%	23%
γ_{ee}	6%	32%
Γ_{ii}	7%	29%
γ_{ii}	8%	31%

5.3.3.3 *MCMC results*

The MCMC was run for $100,000$ iterations, the first $20,000$ were discarded, and the marginal distributions of each parameter were plotted as histograms, Figure 5.6. The Γ_{ee}, Γ_{ii}, γ_{ee}, and γ_{ii} parameters have posterior distributions that are tightly concentrated around the true parameter values. The Γ_{ei}, Γ_{ie}, γ_{ei}, and γ_{ie} parameters have much broader posterior distributions that are highly non-gaussian.

Note that Γ_{ie} has failed to sample close to its true value. This could mean that there is an isolated mode in the parameter space that has not been reached by the MCMC. This may warrant the use of methods for sampling multi-modal distributions such as SMC and population-based MCMC, [Jasra et al., 2007]. As well as isolated modes, the algorithm may be slow to converge because MwG does not take account of the covariance structure in the posterior distribution.

This can be addressed through the use of gradient-based MCMC samplers such as smMALA, which we explore in the following section.

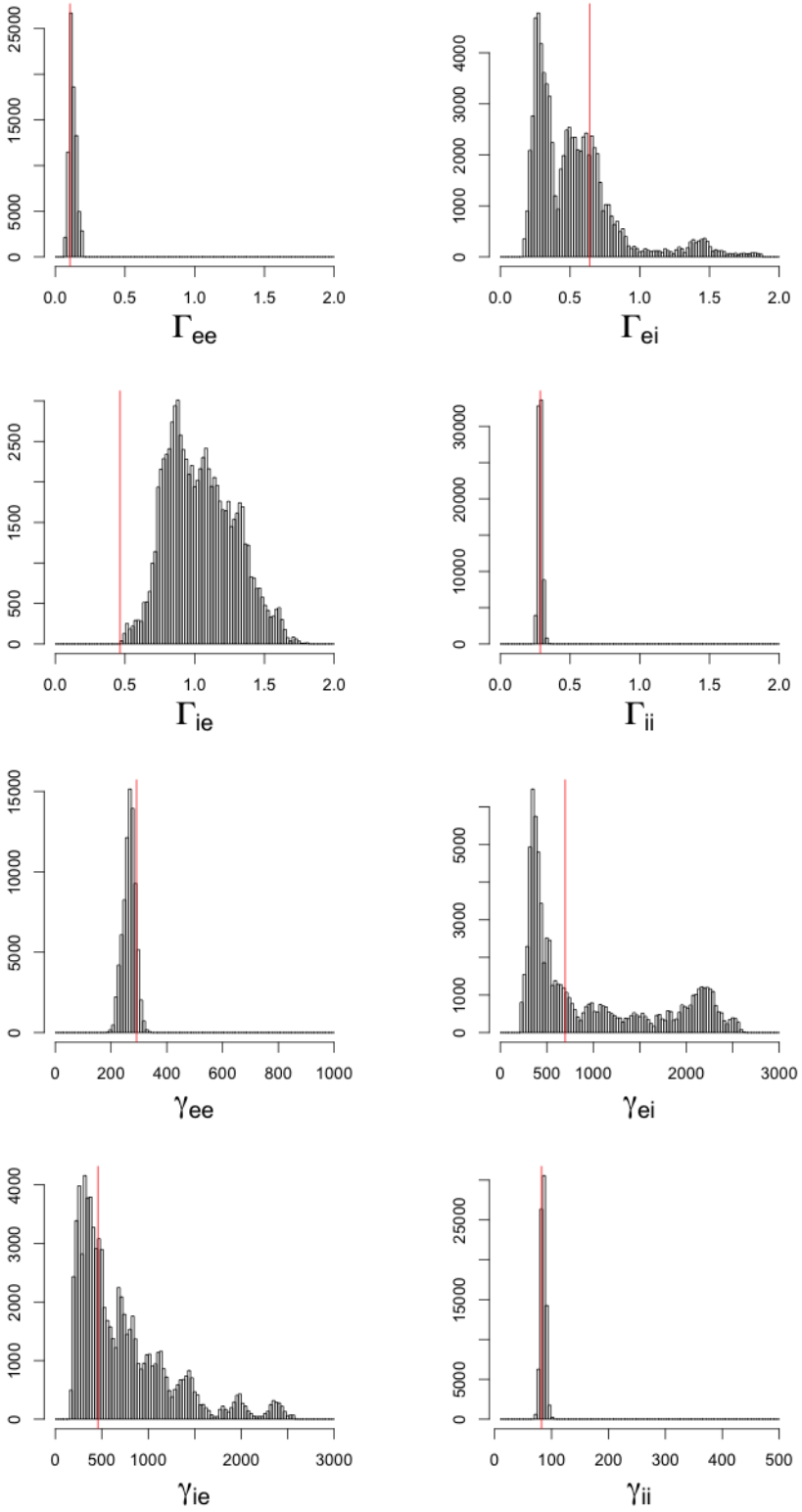

Figure 5.6.: Posterior distributions estimated from MCMC. The vertical red lines indicate the true parameter values, which were used to simulate the data.

5.3.4 *smMALA on stochastic Liley et al model*

In this section we apply the smMALA algorithm to the Liley *et al* model. We could have chosen one of the many other variants of Langevin or Hamiltonian MCMC algorithms to test our methodology, [Girolami and Calderhead, 2011, Hoffman and Gelman, 2014]. The smMALA was chosen because it uses both first and second order derivatives in designing the proposal. This means that it is able to efficiently explore posterior distributions with complex geometries, while still being relatively easy to implement. Other variants of Langevin or Hamiltonian MCMC might perform better than smMALA. We found the smMALA had a tendency to get stuck for a long time in certain regions of the state space. This means that the posterior approximation obtained from smMALA may not be very accurate. Interested readers are referred to the review in [Neal, 2011] for further discussion of sampling efficiency in Hamiltonian and Langevin methods.

NPM knowns		NPM knowns		NPM unknowns		auxiliary	
τ_e	105.5 ms	S_e^{max}	355.2/s	γ_{ee}	841.2/s	sampling frequency	
τ_i	149.0 ms	S_i^{max}	424.8/s	γ_{ei}	859.7/s	f_s	500 Hz
h_e^r	-71.9 mV	$\bar{\mu}_e$	-52.3 mV	γ_{ie}	451.7/s		
h_i^r	-76.0 mV	$\bar{\mu}_i$	-48.6 mV	γ_{ie}	451.7/s	recording duration	
h_{ee}^{eq}	-9.0 mV	$\hat{\sigma}_e$	4.9 mV	q_{ee}	0.9484 $\mu\Omega$C	T	20 s
h_{ei}^{eq}	-10.3 mV	$\hat{\sigma}_i$	3.7 mV	q_{ei}	5.835 $\mu\Omega$C		
h_{ie}^{eq}	-87.5 mV	v	780.7 cm/s	q_{ie}	11.99 $\mu\Omega$C	std. dev. observation noise	
h_{ii}^{eq}	-82.3 mV	Λ	0.345/cm	q_{ii}	11.30 $\mu\Omega$C	σ_{obs}	0.01 mV
N_{ee}^{β}	2194	N_{ee}^{α}	3668	$\bar{p}_{ee}$	6025/s		
N_{ei}^{β}	4752	N_{ei}^{α}	1033	$\bar{p}_{ei}$	1116/s	std. dev. noise input	
N_{ie}^{β}	700					σ_p	100.0/s
N_{ii}^{β}	516						

Table 5.3.: List of NPM parameter values used to generate pseudo-data, taken from column 3 of Table 5 in [Bojak and Liley, 2005], as well as other parameters used in the simulation run. In the inference problem parameters labelled "NPM knowns" are assumed as given, i.e., only the "NPM unknowns" are being inferred from the pseudo-data.

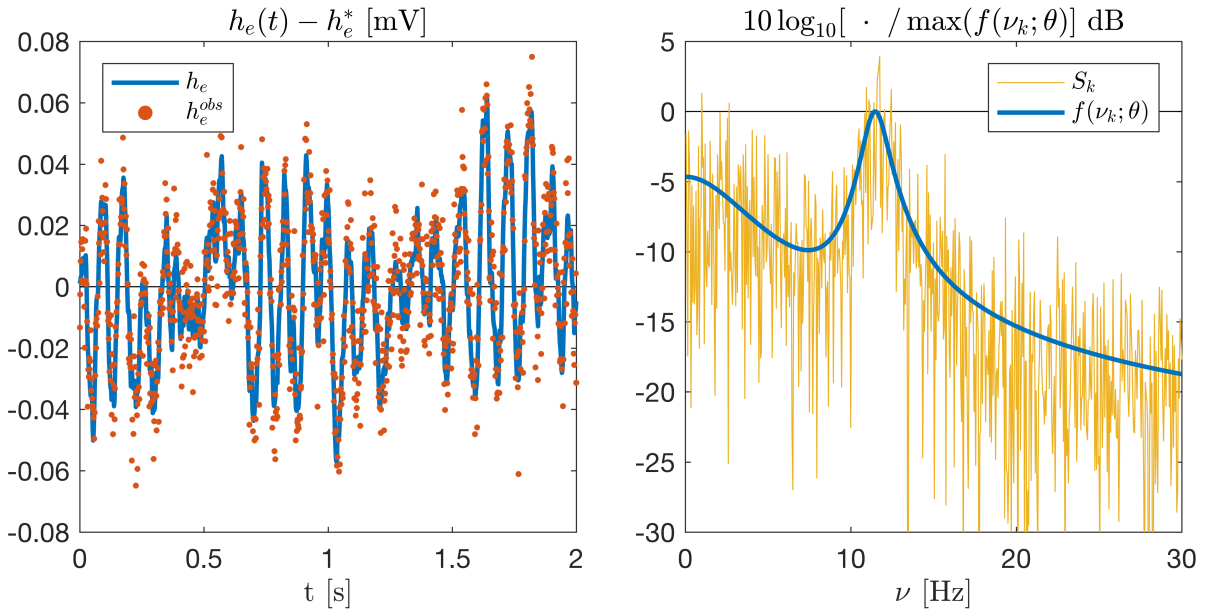

Figure 5.7.: *Left:* First 2 s of $T = 20$ s of pseudo-data generated with parameters from Tab. 5.3 as solid blue line, with noisy observations thereof as red dots. The (equilibrium) mean has been subtracted. *Right:* Decibel plot of spectral density of $T = 20$ s observed pseudo-data (thin golden line) and predicted spectral density of linearized NPM model (thick blue line).

5.3.4.1 *Results*

We chose a product of independent log-normal distributions as the prior distribution for the model parameters. The prior distribution component modes for γ_{ee}, γ_{ei}, γ_{ie}, γ_{ii}, q_{ee}, q_{ei}, q_{ie}, and q_{ii} are shown in Table 5.4. In addition we treated the variance of the input noise as an unknown parameter (denoted by σ^2), also with a lognormal prior distribution. The mode of the prior distribution for this parameter was at 10^8. For most of the parameters we chose the standard deviation of the distribution on the log scale to be 2. For the mean inputs, $\bar{p}_{ee}$ and p_{ei}, the log standard deviation was set to 0.5. Physiologically, the time-varying input should be positive, so putting a relatively tight prior distribution on the mean ensures that this will be the case most of the time.

The prior parameters we have chosen put a lot of prior mass on values above the upper limits in the physiological range. Within the physiological ranges stated in [Bojak and Liley, 2005], the prior density is relatively uniform (it is always at least half of the maximum prior density).

Ideally we may prefer prior distributions that take better account of the physiological ranges. Instead we have found a pragmatic solution that errs on the side of non-informativeness.

γ_{ee}	500	q_{ee}	exp(1)/500
γ_{ei}	500	q_{ei}	exp(1)/500
γ_{ie}	250	q_{ie}	exp(1)/250
γ_{ii}	250	q_{ii}	exp(1)/250
$\bar{p}_{ee}$	5000	p_{ei}	5000

Table 5.4.: Mode of prior distribution components.

The smMALA algorithm was run on data simulated from a known parameter set, referred to as the true parameters. The parameter set used to initialise the MCMC was generated by random sampling with a procedure that was blind to the true parameter values. The algorithm parameters were as follows: Langevin step-size, $h = 0.3$; Number of MCMC iterations $= 100,000$; burn-in length $= 20,000$.

Table 5.5 compares the performance of the smMALA algorithm with the original parameterization against the performance with the reparameterization. The reparameterized model has a lower computational cost per MCMC iteration, and a higher ESS. The cost per iteration is lower because the parameters that are updated deterministically in the reparameterization are an explicit function of the stochastically updated parameters. In the original parameterization the steady-state were defined implicitly, and a nonlinear solver was required to obtain their values given the other parameters. Reparameterizing the model may also have reduced nonlinearities in the likelihood function, leading to the higher ESS. The overall gain in efficiency is a factor of around 10.

The extent of the performance improvement appears to depend on the value of the step-size parameter, h, in smMALA. For example when $h = 0.1$ (instead of 0.3) the two parameterizations have similar acceptance rates. An explanation for this difference is that for the higher h value the discretization of the Langevin diffusion is unstable in a large region of the parameter space for the original parameterization but not for the reparameterized model. In contrast, for the lower

h value the discretization is stable over most of the parameter space in both parameterizations of the model.

Figure 5.8 shows samples obtained from the smMALA algorithm with the reparameterization. The algorithm was initialised from a randomly generated initial parameter set. After around 2,000 iterations, the sampler finds a region of the parameter space with approximately the same likelihood as the true parameters. The algorithm samples close to the true parameter values for some of the time, but the range of parameter sets sampled in regions of high likelihood is relatively wide, indicating weak identifiability and/or non-identifiability of the model parameters.

The samples generated from the smMALA algorithm confirm previous results obtained from sensitivity analysis [Bojak and Liley, 2005], which found that the model is more sensitive to the value of γ_{ii} than to other parameter values, see our Figure 5.8. More interestingly, the samples can be used to identify previously unknown relationships between parameter values in the model. For example, there is strong negative correlation between the h_i steady-state value and q_{ii}, which is proportional to the total charge transferred at synapses between inhibitory neurons, see Figure 5.8.

The results in this section are the first successful application of MCMC methods to estimate parameters in the Liley *et al* model. Indeed we are not aware of any other applications of MCMC to estimate parameters in differential equation models with the same level of complexity as the Liley *et al* model, with the most closely related studies being [Calderhead and Girolami, 2011], [Liepe et al., 2014] and [Penny and Sengupta, 2016]. A relatively common objection to complex models is the lack of identifiability in model parameters. The results here demonstrate that there is some truth in this objection - the marginal posterior distributions for each individual parameter tend to be similar to the prior. However our results also demonstrate that the posterior tends to concentrate towards a subspace with lower dimension than the dimension of the prior. If we are interested in how parameter values change in different conditions (e.g. different levels of anaesthetic) then we can estimate the posterior under each separate condition and see how

	Acceptance rate	CPU time per iteration	Mean ESS	h
Original parameterization	34%	5.66s	36	0.3
Reparameterization	69%	3.18s	239	0.3
Original parameterization	76%	5.66s	23	0.1
Reparameterization	74%	3.18s	34	0.1

Table 5.5.: Comparison of MCMC with and without model reparameterization for smMALA algorithm on NPM. See Sec. 5.3.4.1 for algorithm parameters. The parameter h is the step-size in the smMALA algorithm.

the subspace where posterior density is concentrated changes between conditions. Alternatively, if we have assumptions or a model for how different parameters vary with condition, we can consider the joint likelihood of the parameters given all of the datasets. This may result in a further contraction of the posterior density and is further explored in Chapter 7.

Another important finding of the work in this section relates to the choice of estimation method. Within this chapter we saw that smMALA was more effective than MwG for the Liley *et al* model due to the complex geometry of the posterior distribution. The nonlinearities and non-Gaussianity in the posterior also suggest that the approximate inference methods discussed in Section 2.2 would not be suitable. A more detailed evaluation of approximate inference with sampling based methods is outside the scope of this chapter. However it should be noted that approximate inference methods are increasingly more sophisticated and capable of accurately approximating complex posteriors. See, for example, [Lueckmann et al., 2017].

5.4 DISCUSSION

In this chapter we have developed a way of parameterizing stable differential equation models in terms of their equilibrium. The location of the equilibrium is typically more informative for the model likelihood than the original parameters of the model. Reparameterizing the model makes it easier to sample parameters using MCMC. This can be taken advantage of either by using more basic samplers (such as Metropolis-within-Gibbs), which may be easier to implement and have a lower computational cost per iteration. Or by running more sophisticated samplers for fewer iterations than would otherwise be needed. Sometimes more sophisticated samplers

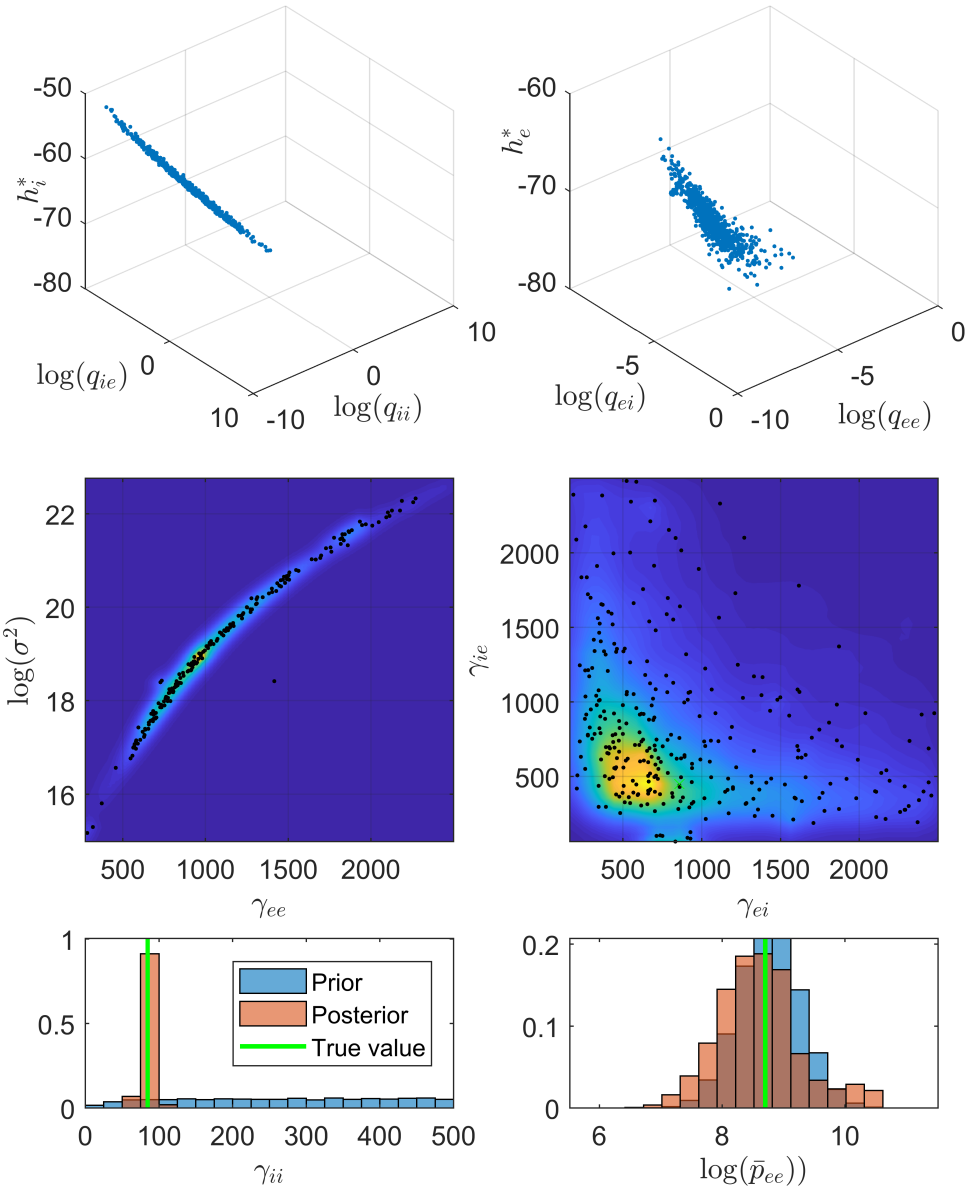

Figure 5.8.: MCMC results with smMALA algorithm for reparameterized Neural Population Model. *Top panels* 3-D scatterplots of every 500th sample (blue dots) for a subset of the parameters, *middle panels* every 500th sample (black dots) for further subset of parameters (same MCMC run as top) and smoothed histogram (background colour map), as in Figure 4.3, *bottom panels*, histograms of MCMC samples for two further parameters and histograms of samples from prior. See Table 5.3 for the true parameter values, and Section 5.3.4.1 for algorithm parameters.

(such as smMALA) are still necessary after reparameterization, but we show that the benefits of reparameterization are retained in this setting as quantified by the Effective Sample Size (ESS).

One area that would be interesting to explore further is the use of ARMA models to re-parameterize stable differential equation models. Linearized stable SDEs can be reduced to ARMA models. It may be possible to parameterize stable SDEs in terms of ARMA coefficients, and we would expect ARMA coefficients to be more informative for the model likelihood than the original parameters. The main challenges in developing this approach would be (i) deciding which parameters get swapped out for the ARMA coefficients and (ii) constructing a map from the reparameterized model to the original parameter space.

More generally, the theory of normal forms for nonlinear dynamical systems can sometimes be used to reduce relatively complex models down to some canonical form. It may be possible to construct a parameterization in terms of the canonical model parameters, and again we would expect this to be more informative for the model likelihood than the original parameters. The challenges would likely be similar to that of the suggested ARMA re-parameterization.

It is also worth considering whether simplified models can be used. In the modelling literature for Computational Neuroscience it is relatively common to construct models so that they have an equilibrium at zero by construction. Constructing models in this way eliminates the need for the re-parameterization method developed in this chapter, but can severely impair the physiological interpretability of the model. Similarly ARMA models and canonical nonlinear dynamical systems are often useful in problems where physiological interpretability is not required because they are much easier to fit than detailed biophysical models.

6

C++ CLASSES AND FUNCTIONS FOR PARAMETER INFERENCE

The R programming language was used to implement and test the methodology developed in Chapters 4 and 5. In order to further develop our work towards data analysis applications we decided to create a fresh implementation from scratch using C++. The benefits of C++ over R are that computation is faster and, within Computational Neuroscience, C++ is more widely used. Another motivation for starting again from scratch was to design the implementation in such a way that minimizes the effort required to change or refine modelling assumptions. In C++ we can define base classes that implement functions common to a pool of potential models. New models can then be implemented by writing a new derived class. Analysis using the new modelling assumptions can then be done by passing an object of the derived type to the likelihood function. This also facilitates comparison of different assumptions on results.

Some of the data analysis applications we are interested in involve fitting a model to multiple datasets. We may want to assume, for example, that some parameters are identical in both datasets. In C++ we can use templates to implement generic algorithms, so that a single implementation of an algorithm can be applied to multiple types of problem. We may also want to run different algorithms (e.g. optimization and MCMC) with the same modelling assumptions.

With these requirements in mind I designed my code with a modular structure, as shown in Figure 6.1. The first module is the data, either a single time-series or multiple time-series that represent recordings in different conditions. The following 3 modules contain the modelling

assumptions, which are separated into (i) assumptions relating to the structure of the model / equations, (ii) a statistical model for the likelihood of parameters given data, (iii) prior distributions representing prior beliefs for parameter values. These 4 modules then feed into the algorithm module. The algorithm produces results, which are written to text files. Figures summarizing the results are produced using a Matlab script that processes the text files generated by the C++.

I implemented an MCMC algorithm (the simplified manifold Metropolis Adjusted Langevin Algorithm, smMALA, discussed in Section 6.3), and an optimization algorithm (conditional maximization, discussed in Section 6.4). Some routes towards extending the range of algorithms are outlined later in this section.

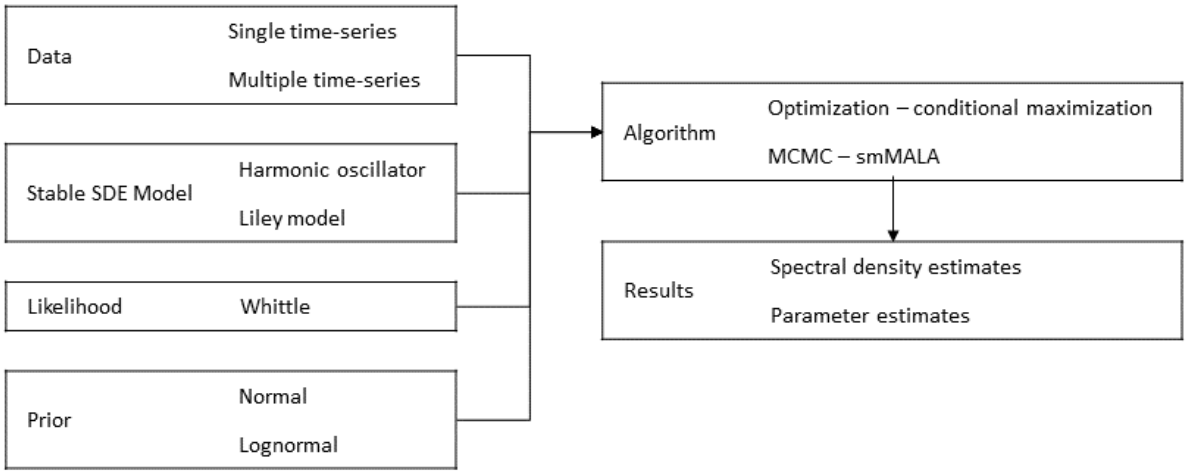

Figure 6.1.: Design for Stable SDE inference code

Uses of optimization

For relatively small and well behaved problems like the harmonic oscillator there is not much use for optimization because we can obtain the same information and more, in terms of accurate uncertainty quantification, by using MCMC. For larger problems MCMC becomes prohibitively expensive. Currently my MCMC sampling takes several days to run for the data analysis problem I am interested in (see Chapter 7 for more details).

One reason MCMC can take a long time is if it is poorly initialized. If there are large negative eigenvalues in the negative Hessian of the target density, the MCMC sampler may take a long time to burn in. Optimization can ameliorate this by identifying parameter sets where the negative Hessian is closer to being positive-definite. See Section 6.3 for further discussion.

The results of optimization can be used to check whether the model can provide a good fit to the data by plotting the predicted spectral density from the model against non-parametric estimates of the spectral density. Running an optimization algorithm that takes seconds or minutes can therefore provide a useful sanity check prior to launching a long MCMC run.

In general I prefer to use prior distributions that err on the side of uninformativity, i.e., that do not fully incorporate prior knowledge. Optimization can then be used to identify whether the posterior has its maximum in a region that is consistent with prior beliefs. It is important to detect this kind of issue because it may point to poor prior assumptions, a poor model (something we discuss further in Chapter 7), or multi-modality in the target density (because the optimization algorithm has failed to find the global maximum).

Addressing multi-modality

The algorithms I have implemented do not handle posterior distributions with multiple modes particularly well. For such posterior distributions I have found that Particle Swarm Optimization (PSO) can identify optimal parameter sets more effectively than either conditional maximization or smMALA. Rather than writing a PSO algorithm myself I chose to write my code in such a way that it is relatively easy to read in the results of PSO (generated using Matlab's `particleswarm` function from the Global Optimization Toolbox). This is not an ideal solution in that it requires models to be coded up twice, once for Matlab and once for C++. And also requires users to have a Matlab licence and the relevant Matlab toolboxes.

An alternative solution to the multi-modality problem that we considered is using Sequential Monte Carlo (SMC), e.g., with the sequence of distributions beginning with the prior and using annealing to get to the posterior [Jasra et al., 2007]. While being relatively simple to implement,

SMC algorithms tend to be more computationally expensive than MCMC. There is more scope for parallelizing SMC algorithms than there is with MCMC. However, to really take advantage of this requires the code to be run on a cluster. I have not installed my code on a cluster and believe that it would take a not inconsiderable amount of time to do this.

Another alternative solution is to use a better optimization algorithm in C++. I considered using functions in the optimization chapter of the NAG library to do this. Similarly to Matlab, the NAG library requires a commercial licence. The University of Reading does have a NAG library licence `http://softwarestore.reading.ac.uk/nag/`, but it is questionable whether other potential users would have such a licence, at least more questionable than whether they would have a Matlab licence. Some effort would also be required to create interfaces to the NAG library routines from my C++ code.

The Stan Math library and the Stan No-U-Turn Sampler (NUTS)

There are a number of connections between my C++ code and the Stan platform for MCMC sampling [Carpenter et al., 2017]. In Stan the four modules that define the input to the algorithm need to be implemented by the Stan user using the Stan Modelling Language. Stan has several interfaces, the most actively used are Rstan and PyStan. The output of the Stan algorithms can be plotted using functions in the interfaces. The algorithms in Stan are implemented in C++ and are hidden from the user. The simplified manifold MALA algorithm is not part of the Stan platform, and more generally the set of algorithms available is fairly restrictive, with the primary MCMC sampler being the No U-Turn Sampler (NUTS). See [Carpenter et al., 2017] for more details.

Another component of the Stan platform is the Stan Math library, which provides a set of building blocks that are used inside the Stan MCMC and optimization algorithms [Carpenter et al., 2015]. The functions in the Stan Math library are templated so that they can either be used with numeric types (such as `double` or `Eigen::VectorXd`), or they can be used with a special Stan type called `var`, which stores information about partial derivatives for automatic

differentiation [Carpenter et al., 2015]. In my C++ I use the Stan Math library for numeric calculations, such as random number generation and evaluating probability densities.

As well as writing in the Stan Modelling language, Stan users can write functions in C++ and use them in their Stan models. This is useful for example, for calculations involving complex numbers (which are not supported in the Stan modelling language), such as the spectral density calculation in Section 4.3. In order to be used by the Stan algorithms these functions need to be templated in such a way that when a `var` is passed as an argument, the partial derivatives are evaluated. It would be possible to interface my C++ spectral density calculations with Stan, but it is unclear whether the time required to implement this would be worth the potential benefits of computational efficiency resulting from using NUTS instead of smMALA. Indeed it is possible that the samples per unit of computational time from NUTS would be lower than with smMALA. This depends on the model, and the geometry of the posterior distribution - see Section 2.3 for a discussion.

Preliminaries for C++ code tutorial

The remainder of the chapter documents my C++ code with the aims of (i) enabling other potential users to implement new models, (ii) demonstrating how I used C++ features such as templated functions, run-time polymorphism and the `std` library to make my code easier to read and maintain. In this chapter the harmonic oscillator is used as a running example, because it has a small number of equations and parameters. However, the code is designed to support MCMC sampling for much more complex models. I have included code snippets from a test called `harmonic-oscillator-example`, which should run and produce results with data supplied through the standard input in the file `harmonic-oscillator-example.d`.

Source code for classes is contained in files with a similar name to the class name. For the example, the header file for the `Parameter_vector` class is contained in `include/parameter-vector.hpp`, and the source code for `Parameter_vector` member functions is contained in `source/parameter-vector.cpp`. Not all of the source code belongs inside classes. For example, `source/`

`bayes.cpp` contains a set of (non-member) functions for evaluating the Whittle likelihood, the prior density, and derivatives of the likelihood and prior.

In terms of prerequisites, potential users should have a good grasp of the C++ standard library, `std`. We make extensive use of the `std` library and include the line `using namespace std;` in all of my source files. An introduction to the standard library can be found in the book C++ Primer [Lippman et al., 2012]. This book also contains a good introduction to modern C++ programming, including features that were added in the C++11 standard.

In addition to this I make use of the `Eigen` library for linear algebra, and thus include the line `using namespace Eigen;` in many of my source files.

6.1 WHITTLE LIKELIHOOD FUNCTION WITH PARAMETER VECTOR INPUT

Section 6.3 presents my implementation of the smMALA algorithm. This section presents the likelihood used by this MCMC sampler, and Section 6.2 presents my implementation of posterior density evaluation. Section 6.4 then discusses how the same posterior density function can be used as an objective function within an optimization algorithm.

6.1.1 *A class for parameter vectors that supports element access by name*

Here I introduce my `Parameter_vector` class. We would like a vector class with the following features.

- Support for standard arithmetic and linear algebra operations. This is useful, for example in using the vector as a parameter in density evaluation and sampling from a multivariate normal distribution.
- The ability to access elements of the vector by parameter name. Especially for complex mechanistic models, it is often much easier to implement formulas such as the system Jacobian when parameter names are used instead of numerical indices.
- Control over copying of parameter values into new and existing objects, for example whether we copy values or copy pointers to values.

Here is an example of how a vector of type `Parameter_vector` can be initialized and accessed.

```
Parameter_vector theta0;
string model_dir = "tests/data/harmonic-oscillator-example/";
ifstream theta_ifstream(model_dir + "up.dat");
theta_ifstream >> theta0; // read theta0 in from file
cout << theta0["omega0"] << endl; // access omega0 by name
cout << theta0[2] << endl;        // access sd_in by numerical index
```

Note that the `>>` operator has been overloaded to enable us to easily read in the data required to initialize a `Parameter_vector` object from a file stream. In the example above, the contents of the file are as follows.

```
omega0      80.0
zeta        0.2
sd_in       100.0
```

In our example the operator `>>` calls a function that iterates over the lines in a file. Each line of the file corresponds to an element in our vector. The last two lines in the example above should therefore print the following.

```
80
100
```

The `<<` operator has also been overloaded, so that the statement `cout << theta0;` prints all elements of the vector with the name of each element. Assuming that we have not modified the vector, this will reproduce the contents of the input file in the standard output.

In the example above `theta0` was initialized without any arguments. Internally this constructor creates a `std::shared_ptr<std::vector<double>>` that points to a vector of length 0. There is also a constructor that compactly initializes a `Parameter_vector` by taking a filename as a string argument, i.e. `Parameter_vector theta0(f_str);`.

In terms of performance, access by index is faster than access by name. Internally access by name is done using an `std::map`, which uses a binary tree search to map names onto indices. (This map is created when the file is read in.) I did some profiling experiments (results not shown) that indicate that access by name may be slower by a factor of around 40. However, I have found that the overall cost of parameter value access is small compared to other computations in my code. And I have found it much easier to debug code that accesses parameters by name

rather than by numerical index. There are some type of computation where the computational cost of parameter access may dominate, e.g. an ODE solver with a small time-step. If this is an issue there is nothing to stop users using numerical indices instead of parameter names when defining a derived class for a given model. It is worth being aware that numerical indices can be defined in an enumeration. This allows users / developers to see parameter names which the C++ preprocessor then converts into integers.

Although not connected with the example above, we now briefly discuss some other ways in which a `Parameter_vector` can be constructed and used. A `Parameter_vector` can be created through a copy constructor, `Parameter_vector theta=theta0;`, which copies the values of `theta0` into `theta`. There are also scenarios where it is useful to have two `Parameter_vector` objects that share access to the same set of values. The expression `p.set_data_ptr(theta);` copies the `std::shared_ptr<std::vector<double>>` of `theta` to `p`. There are also constructors that copy the `std::shared_ptr<std::vector<double>>` so that model-specific functions can access elements of θ. However, I would not expect these constructors to be used outside the context of creating a model object.

Finally, in order to pass a `Parameter_vector` to a function that takes an `Eigen::VectorXd` we have defined a conversion function that copies the values in the `Parameter_vector` to an `Eigen::VectorXd`. And we have overloaded `operator=` to take an `Eigen::VectorXd` and copy it into a `Parameter_vector`.

6.1.2 *A class for data in the time and frequency domains*

Here we are primarily interested in applications where we have time-series data and we would like to analyze that data in the frequency domain. The frequency domain data is obtained by applying a Discrete Fourier Transform (DFT) to the time-series. The `Data_vector` class stores time-series data, transforms time-series data to the frequency domain using the `fftw` library, and stores the resulting periodogram. All these operations can be performed with a single call

to the `Data_vector` constructor, see Section 6.1.3. The data file that is used to initialize the `Data_vector` should have the following form.

```
// time_step        0.01
// time_units       seconds
// freq_lim         0.0     20.0
0.1
0.523964
2.39286
...
```

The first line is used to initialize the time step data member. The second line is used to initialize a constant that is required for scaling the periodogram. The value of this constant depends on the time units of the data (e.g. seconds or milliseconds). The third line defines a range of frequencies for subsetting data in the frequency domain. Subsequent lines of the data input file contain the sequence of time-series observations.

6.1.3 *Evaluating the Whittle likelihood of data given parameters*

I have defined a base class `Stable_sde` that defines functions that are common to the pool of models we are interested in. These functions include the eigen-decomposition and the spectral density calculation. The class also contains pure virtual functions that need to be defined by derived model classes, such as the Jacobian of the SDE drift term.

The Whittle likelihood is then evaluated through the function call operator of an object that contains information about the model and the data. Here is a simplified definition for the Whittle likelihood type,

```
struct Log_whittle_like_ {
  const Data_vector & y_;
  Stable_sde & m_;
  Log_whittle_like_(const Data_vector & y, Stable_sde & m) : y_(y), m_(m) { }

  double operator()(const Parameter_vector & theta) const  {
    // copy theta pointer to p
    m_.p.set_data_ptr(theta);
    // evaluate eigenvectors and eigenvalues of Jacobian
    m_.eigen_dec();
    // evaluate spectral density
    m_.spec_fun(y_.xf, y_.time_step);
```

```
    double ll = 0;
    // evaluate Whittle likelihood from periodogram and spectral density
    for (int k = 0; k < y_.n_freq; ++k){
      ll -= log( m_.spec_dens[k] ) + y_.I[k] / m_.spec_dens[k];
    }
    return ll;
  }
};
```

I have a written a wrapper function for evaluating the Whittle likelihood.

```
double log_whittle_like(const Data_vector & y,
                        Stable_sde & m,
                        const Parameter_vector & theta)
{
  return eval(Log_whittle_like_(y, m), theta);
}
```

The `eval` function calls the `Log_whittle_like_` constructor, then evaluates the log-likelihood at `theta`, conditional on `y` and `m`.

```
double eval(const function<double(const Parameter_vector &)> & callback,
                           const Parameter_vector & x)
{
  return callback(x);
}
```

The reason for creating a callable object (as opposed to just having a function with multiple arguments) is that it makes the calculation of derivatives with finite differences easier, see Section 6.2.4.

Here is an example showing how the `log_whittle_like` function can be used.

```
// initialize Model class
Stable_sde_harmonic_oscillator_a m(theta0, fixed_param);
ifstream m_ifstream( model_dir + "model_input.d" );
m_ifstream >> m;

// read in data
Data_vector y(model_dir + "data1_input.d");

double ll0 = log_whittle_like(y, m, theta0);
cout << "log likelihood at initial parameters = " << ll0 << endl;
```

There a few things to note in this example:

- The `Stable_sde` class has a `Parameter_vector` member that shares data relating to parameter values with `theta0`. This is initialized in the constructor. The second argument

`fixed_param` is another `Parameter_vector` which contains parameters that are set to fixed values and are not modified by algorithms such as MCMC or optimization.

- The `log_whittle_like` function is polymorphic in that any object with a type that is derived from `Stable_sde` can be passed as an argument. In the example `Stable_sde_harmonic_oscillator_a` is derived from `Stable_sde`, and implements the harmonic oscillator model. Details on how this is implemented are provided Section 6.1.4.

The `Stable_sde` class has some data members, e.g., dimension of the state space. We would like to initialize these members from data in a file, and this is implemented by overloading the `>>` operator. Editing this file gives users control over which element of the state vector is observed, and which element has noisy dynamics.

For example,

```
d       2
r_ind   0
l_ind   1
```

This specifies that our model has two state variables. Only the first state variable is observed, and the second state variable has noisy dynamics.

6.1.4 *Defining a derived model class to implement a specific model*

To evaluate the spectral density of a `Stable_sde` we need to evaluate the Jacobian of the drift term in the SDE. Our code is designed to make it easy for users to implement their own derived classes for the models that they are interested in. To evaluate the spectral density in our simple harmonic oscillator example, we define the following member function for the `Stable_sde_harmonic_oscillator_a` class.

```
MatrixXd Stable_sde_harmonic_oscillator_a::sde_jacobian(const Parameter_vector & x) {
  J = MatrixXd::Zero(d, d);
  // define Jacobian for harmonic oscillator
  J(0,0) = 0; J(0,1) = 1;
  J(1,0) = - pow( p["omega0"], 2.0); J(1,1) = -2 * p["zeta"] * p["omega0"];
  return J;
}
```

Elements of the parameter vector are accessed through `p`, which is a `Parameter_vector`.

In addition to `sde_jacobian` users can also define `d_sde_jacobian`. This is required if derivatives of the spectral density are needed.

For more complex models the task of defining these functions can become very labour intensive. We have therefore written a Matlab script to generate code for model specific member functions. This script requires users to define their model inside Matlab. The script then uses this information to,

- evaluate derivatives analytically using Matlab's symbolic toolbox
- convert these derivatives to C code
- copy the C code into a generic template for derived model classes

As a result, users do not need to write any C code themselves in order to define a new derived class for their stable SDE.

6.2 POSTERIOR DENSITY FUNCTION FOR SINGLE AND MULTIPLE TIME-SERIES

Evaluating the posterior density can be a relatively straight-forward product of the prior density and the likelihood function. However, it can become more complex when the prior distribution and posterior distribution have different parameterizations. And in cases where there is not a single likelihood to be evaluated, but multiple datasets, which together define a joint likelihood. I have written my code so that the interface is the same for the simpler and the more complex cases. More specifically, calls to `log_whittle_post` evaluate the posterior density using the Whittle likelihood for all problem types, with problem-specific and model-specific information being defined in derived types that are passed as arguments.

6.2.1 *Evaluating the prior density using a derived class for parameter vectors*

Here we introduce a new type `Parameter_vector_bayes`. This type is derived from `Parameter_vector` and contains additional data that is required for evaluating the prior density.

The >> operator is overloaded for `Parameter_vector_bayes`, so that information about the prior can be read in from a file. For example the file containing information about the prior could be as follows,

```
omega0    lognormal   50.0   20.0
zeta      lognormal   1.0    1.0
sd_in     lognormal   50.0   20.0
```

This sets the prior distribution to be a product of independent log-normal distributions with means equal to the numbers in the third column and standard deviations equal to the numbers in the fourth column. Parameterizing components of the prior by their mean and standard deviation helps make the prior distributions more interpretable. My C++ software is currently restricted to the case where the prior distribution is a product of independent univariate distributions.

The `Parameter_vector_bayes` class defines its own output operator that prints the probability density of each component in the prior as well as the parameter name and the parameter value.

For the purpose of evaluating the posterior density, the `Parameter_vector_bayes` object is stored inside a `Parameter_data` class. The `Parameter_data` class also contains the posterior parameter values and a function to map parameters onto the prior parameterization (more details in Section 6.2.2). The prior information can be read in by passing a file string to the `Parameter_data` constructor, see example in Section 6.2.2.

6.2.2 *Evaluating the posterior density for a single time-series*

It is sometimes desirable to use different parameterizations for the prior density and the posterior density. To implement this we need one function to map parameters from the posterior parameterization to the prior parameterization, and another function to evaluate the log determinant of the Jacobian of this transformation.

In the software these functions are evaluated polymorphically. The base class is `Parameter_data` and the function for evaluating the posterior density is,

```
double log_whittle_post(const Parameter_data & theta_data,
                        const Data_vector & y,
                        Stable_sde & m)
{
  double ld = theta_data.jac_det();
  double ll = log_whittle_like(y, m, theta_data.post);
  double lp = prior(theta_data);
  double lt = ll + lp + ld;
  return lt;
}
```

The `jac_det` method is defined in the `Parameter_data` base class and calls a virtual function `f_jacobian`, which needs to be defined in the derived class.

The `Parameter_data` base class also contains a virtual function `f` to evaluate parameter values on the prior parameterization given posterior parameter values. The function `f` is called inside the `prior` function.

In this simple example the prior and posterior parameter space are the same, so `f` leaves the parameters unchanged and `f_jacobian` is the identity matrix.

Here is an example of how to call `log_whittle_post` with a class that is derived from `Parameter_data`.

```
  string model_dir = "tests/data/harmonic-oscillator-example/";

  // initialize Parameter_data
  Parameter_data_harmonic_oscillator_a theta0(model_dir + "fp.dat",
                                              model_dir + "up.dat",
                                              model_dir + "pp.dat");
  cout << theta0.post;
  cout << theta0.prior;

  // initialize Model class
  Stable_sde_harmonic_oscillator_a m(theta0, model_dir + "initial-state.dat");
  ifstream m_ifstream( model_dir + "model_input.d" );
  m_ifstream >> m;

  // read in data
  Data_vector y(model_dir + "data1_input.d");

  double lt0 = log_whittle_post(theta0, y, m);
  cout << "log posterior at initial parameters = " << lt0 << endl;
```

6.2.3 *Evaluating the posterior density for multiple time-series*

We are interested in evaluating the joint likelihood over multiple datasets. Furthermore, we would like to model the variability of parameter values between datasets. For example, suppose that we wished to fit the harmonic oscillator model to two separate datasets. We could assume, for example, that the values of `omega0` and `sigma_in` are statistically independent between datasets, but that the value of `zeta` is fixed across datasets.

We may also decide to apply a log transformation to the parameters in the posterior parameter space. As a result we need to re-define the derived `Parameter_data` class so that prior densities are evaluated on the original parameter space and the Jacobian of the log transformation is evaluated. Somewhat less obviously we also need to re-define the derived `Stable_sde` class so that, for example, the SDE system Jacobian is expressed in terms of the log-transformed variables.

For this problem `theta0` might contain the following parameter values,

```
log_omega0_c1     4.4
log_omega0_c2     3.7
log_sd_in_c1      3.5
log_sd_in_c2      2.3
log_zeta          -1.6
```

where the suffixes `_c1` and `_c2` differentiate the value of a parameter in different conditions.

In order to evaluate the joint likelihood we need a way of evaluating the Jacobian of the SDE for each dataset. In our example, the same Jacobian function is used for each dataset, but the value of the Jacobian is different for different datasets because of the variability in ω_0 and σ_{in}.

The model for the joint likelihood is implemented as a `vector< shared_ptr<Stable_sde> >`, i.e. a vector of pointers to `Stable_sde` objects.

The table below illustrates how parameter values are referred to inside the `Stable_sde` vector.

The reason we have to work with `Stable_sde` pointers is an idiosyncrasy of C++: it is not possible for a `vector< Stable_sde >` reference to refer to a `vector< Stable_sde_simple >`. But it is possible for this kind of dynamic typing to be performed using pointers.

`Parameter_vector` interface	`Stable_sde` interface	Value
`theta0["log_omega0_c1"]`	`(*m[0]).p["log_omega0"]`	4.4
`theta0["log_omega0_c2"]`	`(*m[1]).p["log_omega0"]`	3.7
`theta0["log_sd_in_c1"]`	`(*m[0]).p["log_sd_in"]`	3.5
`theta0["log_sd_in_c2"]`	`(*m[1]).p["log_sd_in"]`	2.3
`theta0["log_zeta"]`	`(*m[0]).p["log_zeta"]`	-1.6
-	`(*m[1]).p["log_zeta"]`	-1.6

Table 6.1.: Interface for accessing `theta0` from inside model.

The map from the `p` parameter names to numerical indices of `theta` is user-defined. For example, here is the file input that created the name-index map for `(*m[0]).p`,

```
log_omega0        log_omega0_c1
log_zeta          log_zeta
log_sd_in         log_sd_in_c1
```

Defining the submodel maps externally to the C++ program makes it easy to modify assumptions relating to parameter variability without having to recompile the code.

We can initialize the objects containing the parameters, the data and the model, and then evaluate the posterior density as follows.

```
string model_dir = "tests/data/harmonic-oscillator-example/";
string res_dir = "tests/results/harmonic-oscillator-example/";

// initialize Parameter_data
Parameter_data_harmonic_oscillator_b theta0_data(model_dir + "fp.dat",
                                                 model_dir + "up_multi_ds.dat",
                                                 model_dir + "pp_multi_ds.dat");
cout << theta0_data.post;
cout << theta0_data.prior;
Parameter_data_harmonic_oscillator_b theta_data = theta0_data;

// initialize submodels
Stable_sde_multi_ds m;
vector<string> fstr_vec = {model_dir + "submodel1_map.d",
                           model_dir + "submodel2_map.d"};
for (string fstr : fstr_vec ){
  // initialize submodel with different parameter map for each submodel
  m.push_back(make_shared< Stable_sde_harmonic_oscillator_b >(
    theta0_data,
    model_dir + "model_input.d",
    fstr,
    model_dir + "initial-state.dat" ) );
}

// read in data
```

```
Data_vector_multi_ds y;
fstr_vec = {model_dir + "data1_input.d", model_dir + "data2_input.d" };
for (string fstr : fstr_vec){
  y.push_back( make_shared<Data_vector>(fstr) );
}

double lt0 = log_whittle_post(theta0_data, y, m);
cout << "log posterior at initial parameters = " << lt0 << endl;
```

The most complex part of this code is the call to the `Stable_sde_harmonic_oscillator_b` constructor. The constructor initializes a `Parameter_vector`, `p` that can access `theta0` with the interface defined in Table 6.1. Other data members for the model are initialized from values provided in the input file, as described above. The operations of the constructor are not specific to the derived class, so all the work is done in the base class (`Stable_sde`) constructor, which is called by the derived class constructor.

In the code, the `Stable_sde_harmonic_oscillator_b` constructor is called through the standard library `make_shared` function. The `make_shared` function returns a `shared_ptr` to a dynamically allocated `Stable_sde_harmonic_oscillator_b` object. In this example, the only differences between different elements of the model vector are in the map that determines which elements of θ are used by each sub-model.

Another point worth noting is that we also use pointers to access `Data_vector` objects. This is not strictly necessary. However, it makes the style consistent between the model and the data objects. And it makes it relatively easy to add polymorphism to `Data_vector` objects at a later point if we decided that was useful.

The `log_whittle_post` function is overloaded so that when multiple datasets and a vector of `Stable_sde` pointers are passed, the joint log likelihood is evaluated by looping over datasets, adding the log-likelihood for each dataset to the joint likelihood on each iteration.

6.2.4 *Evaluating derivatives of the posterior density*

The derivatives of the posterior density can be evaluated using finite differences.

```
Parameter_vector grad_lt0 = log_whittle_post_grad(theta0, y, m, dat.fd_step_size);
MatrixXd hess_lt0 = log_whittle_post_hess(theta0, y, m, dat.fd_step_size);
```

We may want to evaluate the derivatives of other functions. For this reason I wrote a function, `grad_fd`, that calculates finite difference derivatives using the standard library `function` type. Both functions and classes that overload the function-call operator can be passed to `grad_fd`. Finite difference approximations to the prior density, Jacobian determinant and the log likelihood can all be calculated using this same function. This would not be possible had the `callback` argument been a function pointer.

```
Parameter_vector grad_fd(const function<double(const Parameter_vector &)> & callback,
                         const Parameter_vector & x0, const double & epsilon )
{
  double fx0 = callback(x0);
  Parameter_vector fx_grad( x0.get_names(), 0.0);
  for (int i = 0; i < x0.size(); ++i){
    Parameter_vector x = x0;
    x[i] += epsilon;
    double fx = callback(x);
    fx_grad[i] = (fx - fx0) / epsilon;
  }
  return fx_grad;
}
```

An example of a call to this function is as follows,

```
Parameter_vector ll_grad = grad_fd( Log_whittle_like_( y, m ), theta, fd_step_size  );
```

This statement calls the `Log_whittle_like_` constructor passing in the data, `y`, and the model, `m`. When the program reaches `callback` inside `grad_fd` the function-call operator is called and the log likelihood is evaluated.

6.3 SAMPLING FROM THE POSTERIOR DENSITY WITH MCMC

The MCMC sampler is implemented in a template class, `MH_move_smMALA<T1,T2>`. The template parameters are the type of dataset (single or multiple time-series) and the type of model. The class member for the model is a reference, so `MH_move_smMALA` objects can sample the parameters of any model derived from `Stable_sde` and the templating means that the model member can be a single model or a vector of pointers to models, as described above.

The MCMC parameters, such as number of iterations and step-size are read in from standard input, into a struct called `dat`. This struct is then passed to the `MH_move_smMALA` constructor. The output operator `<<` is overloaded so that diagnostic information (such as the

current value of the likelihood) is printed to the standard output. We are interested in obtaining samples from the posterior and this information can be sent to a separate results file (`mcmc_samples.dat` in the example below). Furthermore, we are interested in predictions of the spectral density, and these predictions can also be written to file (`"spec_dens_c1_samples.dat"` and `"spec_dens_c2_samples.dat"` in the example below).

The MCMC moves are implemented in the function-call operator for `MH_Move_smMALA`. The arguments of this operator are references to `Parameter_data` objects. This enables the sampler to be polymorphic in that any class derived from the `Parameter_data` base class can be passed as an argument. And by making the `Parameter_data` objects arguments of the function-call operator (as opposed to class members) we can access and print information about the parameters more easily.

Here is an example of how to call the MCMC sampler for a problem with multiple datasets.

```
// read MCMC and problem parameters from standard input
Std_input dat;
cin >> dat;

MH_move_smMALA_multi_ds mh_move_smMALA(theta0_data, y, m, dat, VERBOSE);
cout << mh_move_smMALA << endl;

// initialize an output stream for parameter names and values to a file
ofstream ofs( res_dir + "mcmc_samples.dat" );
ofs << theta0_data.prior.get_names() << endl;
ofstream ofs_spec_dens_c1( res_dir + "spec_dens_c1_samples.dat" );
write_row_vec( ofs_spec_dens_c1, (*y[0]).get_freq() );
ofstream ofs_spec_dens_c2( res_dir + "spec_dens_c2_samples.dat" );
write_row_vec( ofs_spec_dens_c2, (*y[1]).get_freq() );

for (int i=1; i < dat.n_mcmc_iter; ++i){
  // do MCMC move and write diagnostics to standard output
  cout <<  mh_move_smMALA(theta0_data, theta_data) << endl;

  // write parameters to file
  theta0_data.prior.write_row(ofs);
  // write spectral density to file
  write_row_vec(ofs_spec_dens_c1, (*m[0]).get_spec_dens() );
  write_row_vec(ofs_spec_dens_c2, (*m[1]).get_spec_dens() );
}
```

Here is a simplified version of the function-call operator that implements a Metropolis-Hastings update with the smMALA algorithm.

```
MH_move_smMALA<T1,T2> & MH_move_smMALA<T1,T2>::operator()(Parameter_data & theta0_data,
                                                          Parameter_data & theta_data)
{
  Parameter_vector & theta0 = theta0_data.post;
  Parameter_vector & theta  = theta_data.post;

  // sample parameters - assigns values of Parameter_vector using VectorXd
  theta = stan::math::multi_normal_rng(mu0, C0, gen);
  theta_data.set_prior_vals(); // update values of parameter values in prior space

  // evaluate likelihood
  lt = log_whittle_post(theta_data, y, m);
  lt_diff = lt - lt0;

  // compute mean and covariance of proposal
  Parameter_vector grad = log_whittle_post_grad(theta_data, y, m, fd_step_size);
  MatrixXd hess_lt = log_whittle_post_hess(theta_data, y, m, fd_step_size);
  MatrixXd C = cov(hess_lt);
  VectorXd mu = theta + 0.5 * C * grad;

  // evaluate pdf of proposals
  double q_theta0 = stan::math::multi_normal_log(static_cast<VectorXd>(theta0), mu, C);
  double q_theta =  stan::math::multi_normal_log(static_cast<VectorXd>(theta), mu0, C0);
  q_diff = q_theta0 - q_theta;

  aprob = min(1.0, exp( lt_diff + q_diff ) );
  if ( aprob > stan::math::uniform_rng(0,1,gen) ){
    theta0_data = theta_data;
    lt0 = lt; mu0 = mu; C0 = C;
  }
}
```

Here is the terminal output from an MCMC run that estimates the parameters of the harmonic oscillator model across multiple datasets. The data is generated from the model and the parameters are initialized at their true values.

```
it       lt0            lt_diff         q_diff          min_lambda      acc_rate
0        6150.87        -               -               141.018         -
100      6158.06        -1.34164        1.11568         126.781         0.75
200      6158.22        4.36964         -3.94354        124.58          0.69
300      6151.64        -6.51176        4.99062         113.507         0.79
400      6158.01        0.227468        -0.109221       139.028         0.75
500      6157.35        -0.769251       0.571616        125.421         0.82
600      6156.51        1.11926         -0.737413       133.515         0.77
700      6153.98        -2.59266        2.10324         158.08          0.79
800      6152.88        -1.96021        1.88233         103.647         0.75
```

```
900       6157.55          4.87941          -3.49269          128.412          0.7
1000      6158.62          -2.06967         1.58933           124.893          0.76
```

These results were generated with the following input parameters,

```
1E-4     // step-size for finite differences
1.0      // step-size in smMALA algorithm
1000     // number of MCMC iterations
soft_abs // method for regularizing Hessian
1E6      // alpha parameter used for regularizing Hessian using soft_abs
100      // stride for printing
```

In addition the observation noise is treated as a fixed known parameter, and is equal to 0.05.

The seed for the random number generator is initialized using the `time` function so different runs will give different results. However, the acceptance rate (which is an average over `stride` iterations) should be similar from run to run. In this case the acceptance rate is high and the chain appears to be mixing well. The target density values `lt0` should be similar from run to run once the chain is burnt in, independently of where in the parameter space the sampler is initialized from.

If the chain is not mixing well the other columns in the terminal output should give some indication as to why this is the case. Firstly the likelihood and target density at the proposed parameter set should not be a lot less than the target density (e.g. `lt_diff < -10.0` is a sign of a poor proposal distribution). Also if the acceptance rate is very high, but `lt_diff` is very small (e.g. `abs(lt_diff) < 0.01`), this is a sign that the MCMC sampler is exploring the parameter space very slowly (i.e. it is mixing poorly).

It is sometimes the case that `lt_diff` is ok but `q_diff` (the log ratio of the proposal density ratio) is negative and outweighs `lt_diff` leading to a low acceptance rate. The smMALA algorithm adapts the variance of the proposal distribution based on the local geometry of the posterior distribution. If the proposal variance is small at the current parameter set, θ_0, but large at the proposed parameter set, θ, this could lead to smaller values of $q(\theta_0|\theta)/q(\theta|\theta_0)$. There is not much that can be done about this, except perhaps to use a different MCMC algorithm, or run the sampler for a very long time to make up for the poor mixing.

The regularization of the target density Hessian can also lead to poor mixing. The quantity `min_lambda` is the lowest eigenvalue of the negative Hessian, $-H(\theta)$. Whenever $-H(\theta)$ has negative eigenvalues, I regularize the Hessian using the method described in [Betancourt, 2013]. If `min_lambda` is large and negative, then a lot of regularization is needed in order to obtain a valid proposal distribution (all the eigenvalues of the proposal covariance must be positive). When $-H(\theta)$, has a large negative eigenvalue, the regularized matrix will have a new large positive eigenvalue. This will lead to a smaller variances in the proposal distribution. The effect of this is difficult to predict, but generally speaking the smMALA sampler does not mix as well when it is initialized in regions of the parameter space where $-H(\theta)$ has large negative eigenvalues.

One solution to this problem (presented in Section 6.4) that I have found to be effective is to optimize the target density and then start the MCMC sampler from the optimal parameters. At a maximum of a multivariate function all the eigenvalues of $-H(\theta)$ are positive. If the derivatives are relatively smooth around some neighbourhood of the maximum, we would expect that not much regularization is required during MCMC sampling.

Some care is required in choosing an optimization algorithm as some commonly used optimization algorithms, such as Newton's method, are not very effective when the objective function is not positive definite (i.e. when there are negative eigenvalues).

6.3.1 *Plotting and summarizing MCMC results*

Another check for good MCMC mixing is that when the parameters are estimated from synthetic data (i.e. when the actual underlying parameter values are known) the estimates are consistent with the actual parameter values. This section presents some results for the harmonic oscillator model fitted to a pair of datasets that were generated by simulating from the model. In Table 6.2 we show that the 95% credible intervals for each parameter include the actual parameter values for all 5 parameters. These results came from one MCMC run of $10,000$ iterations.

	actual	0.025	0.50	0.975
$\omega_0(c_1)$	80	77.3	80.3	81.9
$\omega_0(c_2)$	40	36.8	38.8	41.3
$\sigma_{in}(c_1)$	100	92	101	111
$\sigma_{in}(c_2)$	10	9.81	10.8	12.4
ζ	0.2	0.164	0.193	0.223

Table 6.2.: Estimated quantiles of posterior distribution from MCMC

In the absence of actual parameters from the data generating process it is a good idea to check that the spectral density predictions generated by sampled parameter sets are consistent with non-parametric estimates of the spectral density. Figure 6.2 shows 95% confidence intervals for the spectral density obtained using the Welch method alongside 95% credible intervals for the spectral density estimated from $10,000$ MCMC iterations.

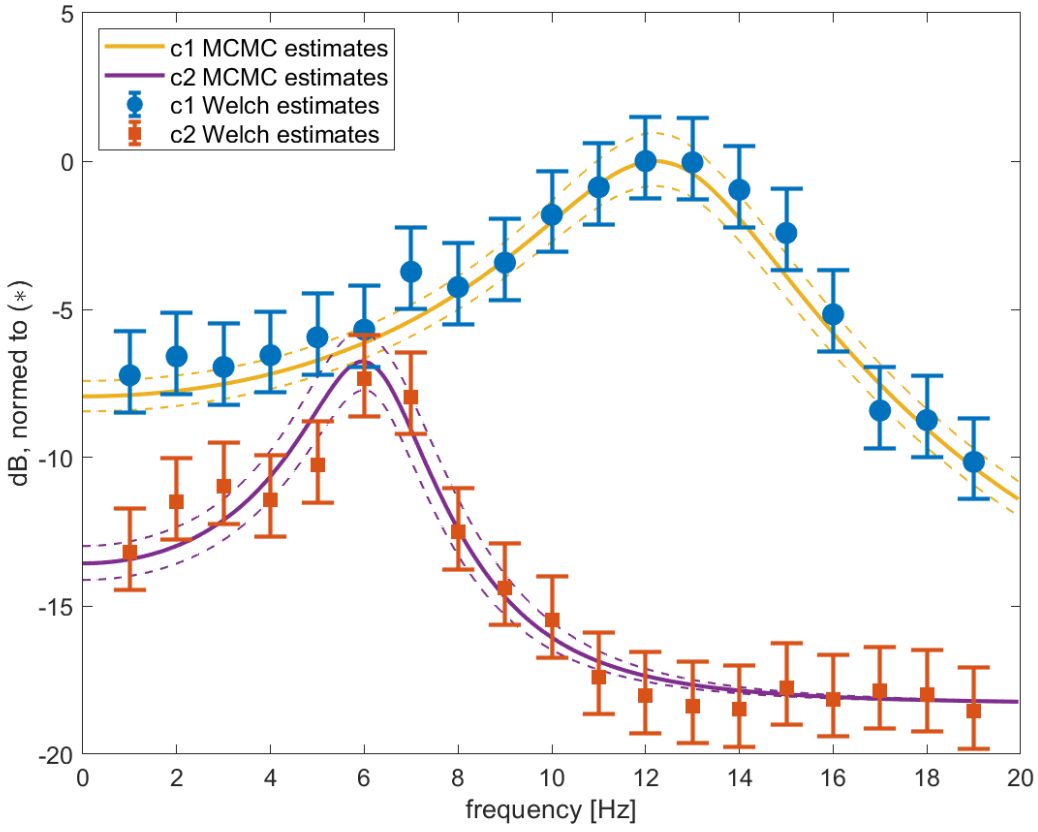

Figure 6.2.: Spectral density estimates for harmonic oscillator model from synthetic data

6.4 MAXIMIZATION OF THE POSTERIOR DENSITY

Similarly to the MCMC sampler I have implemented my optimization algorithm in a template class, `Opt_move<T1,T2>`, where the template parameters are the type of dataset and the type of model. I chose to implement conditional maximization, informed by Chapter 13 of [Gelman et al., 2014]. In conditional maximization the parameters are updated one at a time. For each parameter a local quadratic approximation to the conditional target density is constructed. The parameter is updated by maximizing the local quadratic approximation. This algorithm is straight-forward to implement, does not require any tuning parameters to be set, and is effective in cases where $-H(\theta)$ is not positive definite. One drawback is that it is slow to converge, especially when there are strong correlations between parameters in the target density. However, for the purposes of evaluating how well a model can fit a dataset, and for finding regions of the parameter space that are a good starting point for smMALA, accurate identification of the optimal parameter set is not required.

Conditional maximization has a lower computational cost per iteration than smMALA because in conditional maximization we only need the diagonal elements of the Hessian, whereas the smMALA computes a full Hessian on each iteration.

Another drawback that poses a more serious problem is that the algorithm is designed to converge toward local modes. If the target density is multi-modal the algorithm should be run with multiple initial conditions. For some multi-modal problems we have found that other algorithms, such as Matlab's `particleswarm` function, can be more effective.

The initial condition currently needs to be specified in an input file. Providing the input in this way makes it easier to use results generated by other codes. This is something I have used to initialize my MCMC sampler for the Liley *et al* model at a parameter set identified by Matlab's `particleswarm`. I also found it useful to specify parameter values in an input file for validating my C++ code against my R code.

The conditional optimization algorithm can be called as follows.

```
  Opt_move_multi_ds opt_move(y, m);
  int i = 1;
  while (opt_move.get_abs_diff() > 1E-3 & i < 100){
    cout << opt_move(theta0_data, theta_data, 1E-4) << endl;
    ++i;
  }
  // write parameters to file
  ofstream ofs( res_dir + "opt_multi_ds.dat" );
  ofs << theta0_data.post << endl;
```

On each iteration the absolute difference in the log target density is calculated for each update. The method `get_abs_diff()` returns the maximum of the differences across all parameter updates. In the example above, the algorithm stops if this quantity is less than 10^{-3}. And we have set the maximum number of iterations to be 100. Again this is because we are primarily interested in reaching an area of high posterior density relatively quickly.

Here is the function-call operator that implements the conditional maximization update.

```
template <class T1, class T2>
Opt_move<T1,T2> & Opt_move<T1,T2>::operator()(Parameter_data & theta0_data,
                                              Parameter_data & theta_data,
                                              double fd_step_size)
{

  Parameter_vector & theta0 = theta0_data.post;
  Parameter_vector & theta  = theta_data.post;

  double lt0 = log_whittle_post.lt;
  Eigen::VectorXd abs_diff_vec = VectorXd::Zero( theta0_data.post.size() );
  for (int i=0; i < theta0_data.post.size(); ++i){
    theta_data = theta0_data;

    double grad_lt0 = log_whittle_post_grad(theta0_data, y, m, fd_step_size, i);
    double hess_lt0 = log_whittle_post_hess(theta0_data, y, m, fd_step_size, i);

    if ( ( !isinf(grad_lt0) ) && (grad_lt0 != 0) && (hess_lt0 > 0)  ){
      theta[i] = theta0[i] + grad_lt0 / hess_lt0;
    }
    theta_data.set_prior_vals(); // update values of parameter values in prior space

    //  evaluate likelihood at new parameters
    double lt_star = log_whittle_post(theta_data, y, m);
    if ( lt_star > lt0 ){
      theta0_data = theta_data;
      abs_diff_vec[i] = lt_star - lt0;
      lt0 = lt_star;
    }
  }
```

```
  abs_diff = abs_diff_vec.maxCoeff();
}
```

Here is the terminal output from running conditional maximization of the posterior density across multiple datasets for parameters of the harmonic oscillator model. The data is generated from the model and the parameters are initialized at their true values.

it	lp	ld	ll	lt	abs_diff
0	-26.1819	15.7	6161.36	6150.87	-
1	-25.5506	15.6456	6168.65	6158.75	4.1361
2	-25.4803	15.656	6168.87	6159.05	0.2138
3	-25.4655	15.6524	6168.87	6159.06	0.0070
4	-25.4625	15.6486	6168.87	6159.06	0.0009

Note that in overloading the output operator for `Opt_move` objects we have decomposed the log target density into its constituent parts - the prior, `lp`, the Jacobian determinant, `ld` and the likelihood, `ll`.

6.5 DISCUSSION

In this chapter I have given an overview of my C++ implementation for parameter estimation in stable SDEs using the Whittle likelihood. The chapter should provide a useful introduction to what I have programmed and why for anyone who wishes to familiarize themselves with my code.

The motivation behind the work in this chapter was to write code that could more easily be shared, and used as a piece of software for other similar research problems. There are a number of things that could be done to further this objective, for example setting up a public repository for the code and providing a set of installation instructions. Installing and compiling C++ code can be tricky, so it would also be interesting to explore whether the code could be accessed in a Docker container (`https://blogs.msdn.microsoft.com/vcblog/2018/08/14/c-development-with-docker-containers-in-visual-studio-code/`) which would enable users to interact with the software without having to install it on their machine.

A further area for software development would be to enable my software to be called from other programming languages, such as Matlab, Python and R. All these languages have tools

that enable code to be written in C++ and called from the higher-level language (mex for Matlab, ctypes for Python, and RCpp for R). Using these tools for my code may not be entirely trivial because of the advanced C++ features that I have used, such as inheritance, and my customized class for parameter vectors. Being able to do this would make it easier to experiment with different algorithms and to plot results.

7

EEG DATA ANALYSIS

In preceding chapters we have restricted our attention to cases where the data is synthetic, i.e., it is generated by simulating from the model. The advantage of doing this is that we can assess the performance of our Bayesian estimation algorithms without having to worry about the effect of model mis-specification. Ultimately, however, the methods we have developed are only useful if the underlying Neural Population Models are a reasonable approximation to the processes that generate neural activity. And furthermore that our model of the relationship between underlying neural activity and observations is accurate. The aim of this chapter is to demonstrate that this is indeed the case. In Section 7.1 we fit the simple E-I network model from Section 3.1.4 to human adult resting state EEG data. Then in following sections we fit the Liley *et al* model to rat EEG data from experiments at several levels of anaesthesia with the isoflurane agent. Whereas previous studies have estimated the parameters of simpler NPM models, [Moran et al., 2009, Abeysuriya and Robinson, 2016, Hashemi et al., 2018], the results in this chapter are novel in doing this for a more complex model, and with more accurate uncertainty quantification. The results are also novel in being the first use of the Whittle likelihood for estimating the parameters of NPMs from EEG data.

In contrast to the simple E-I network model, the Liley *et al* model (which was also introduced in Section 3.1.4) is a more interpretable and detailed model of brain physiology. Furthermore, independent experiments have identified physiological ranges for these parameters. By incorpo-

rating these ranges into the prior distribution for the parameters we can assess the accuracy of the model by evaluating whether good fits to the data can be obtained with physiological parameter values.

In the case where there are multiple datasets, priors can also be used to specify the relationship between parameters across datasets. In preliminary work we tried treating parameters as independent across datasets. However, the result of this was that parameters which we were fairly confident, based on prior knowledge, should not change between datasets did change. Previous studies have addressed this issue by fitting in multiple stages, e.g., obtain parameter sets that fit the data well in one condition then allow a subset of the parameters to vary with anaesthesia level. However, this approach can be inefficient in that there may be many parameter sets that fit the data well in one condition but not in others. This lead me to develop a different approach that involves sampling from the posterior distribution of the parameters given all of the data, but with some of the parameters constrained to be identical across datasets. A simple example of this was given in the previous chapter with the harmonic oscillator. On one level this chapter simply applies the ideas of the previous chapter to a more complicated model, and to experimental data instead of synthetic data. Whereas the previous chapter was written to give an accessible introduction to the software I have written, in this chapter we are more focused on generating insights into the mechanisms underlying the changes in neural activity that occur during anaesthesia in response to isoflurane.

7.1 ANALYSIS OF ADULT RESTING STATE EEG DATA

Relatively early on in the PhD project we tested our statistical methodology on some experimental data. We obtained some adult resting-state EEG data from our collaborator, David Liley.

At that time the development of our statistical methodology was not sufficiently advanced to estimate the parameters of the Liley *et al* model, but we found that we were able to get good results, in the sense of being able to fit the spectrum well with a simple E-I network model.

This model was discussed in Section 3.1.4. We briefly reiterate the model equations here for convenience.

$$\frac{d^2h_I}{dt^2} + 2\zeta\omega_0 \frac{dh_I}{dt} + \omega_0^2 h_I = \alpha h_E \tag{7.1}$$

$$\frac{dh_E}{dt} = -\beta h_I + \gamma h_E + p(t), \tag{7.2}$$

Observations are assumed to be a linear function of $h_E(t)$ with some observation noise, and the function $p(t)$ is assumed to be a white noise process with variance σ_p^2.

I used a Metropolis-within-Gibbs MCMC sampler (see Section 2.3.2 for a description) to estimate the posterior distribution of the parameters given the EEG data. The Whittle likelihood was evaluated with frequencies in the range $1 - 40$Hz.

The results of the MCMC for each parameter are summarised in Table 7.1. We can see, for example, that the strength of recurrent excitatory feedback, γ, is much smaller than the other feedbacks, ω_0, α and β.

In the absence of coupling with the excitatory population the peak in the spectral density would be at $\omega_0\sqrt{1-\zeta^2}$. In terms of the posterior distribution of ω_0 and ζ, that would be at around 50Hz. The excitatory coupling causes this peak to shift down to around 9Hz for this dataset. See Figure 7.1.

Figure 7.2 shows that the α and β parameters are negatively correlated with each other in the posterior distribution. Low α and high β produces similar behaviour to high α and low β. This type of the correlation in the posterior / lack of identifiability frequently occurs with complex mechanistic models. For this simple model it is possible to show algebraically that the likelihood is invariant to the value of $\alpha\beta$, e.g., parameters sets with $\alpha = 2,\ \beta = 3$ will have the same likelihood as $\alpha = 12,\ \beta = 0.5$ since $\alpha\beta = 6$ in both cases. To see this, first eliminate $h_I(t)$ from the model since it is not observed. To do this re-arrange Equation (7.2) in terms of βh_I, differentiate twice to get expressions for the derivatives of h_I, multiply Equation (7.1) through by β then substitute out βh_I and its derivatives. The result is a 3rd order differential equation.

Parameter	Posterior mean	Posterior 95% credible interval
ω_0	60.4	(59.1, 62.0)
ζ	0.396	(0.356, 0.437)
γ	2.02	(0.211, 4.73)
α	229	(168, 357)
β	660	(402, 900)
σ_p^2	57.3	(53.7, 61.0)

Table 7.1.: MCMC results for simple model

The expression $\alpha\beta$ appears in the coefficient of h_E, and α and β do not appear in any of the other coefficients. This is consistent with the observation that in Figure 7.2 that posterior is concentrated close to the curve $\alpha\beta = c$, with $c \approx 1.5 \cdot 10^5$.

The results for this model also indicate why the MwG sampler is unlikely to scale well to more complex problems. The MwG sampler proposes parameters parallel to the coordinate axes. So if there is a strong correlation between parameters in the posterior then MwG will tend to have a low acceptance rate and/or move slowly around the parameter space. Generally speaking we would expect this problem to become more pronounced when there are more state variables and more unknown parameters with complex dependencies. And as more data is collected the posterior will become more concentrated around the curves / subspaces with high posterior density.

I did some preliminary work on fitting the simple E-I network to the rat anaesthesia data that is analyzed in the following section. However, I found that I was not able to get a good fit to the spectrum. This appeared to be a limitation of the model rather than an inability of the fitting algorithm to find the optimal parameters.

7.2 DEVELOPMENT OF PRIOR DISTRIBUTION FOR NPM PARAMETERS

In this section and following sections, our focus is on inference using the Liley *et al* NPM model. The first prior distribution for the parameters of this model that was tried was a uniform distribution using the ranges specified in [Bojak and Liley, 2005]. The main problem that I encountered when using this prior was that my MCMC sampling algorithms had a tendency

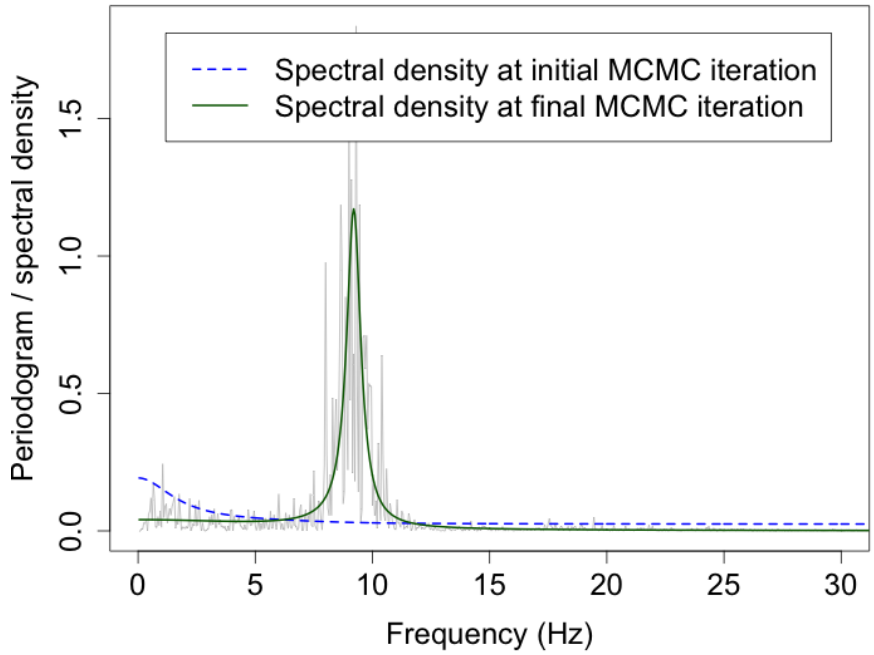

Figure 7.1.: Data in frequency domain (grey) and predicted spectral density for 2 different parameter sets (blue and green).

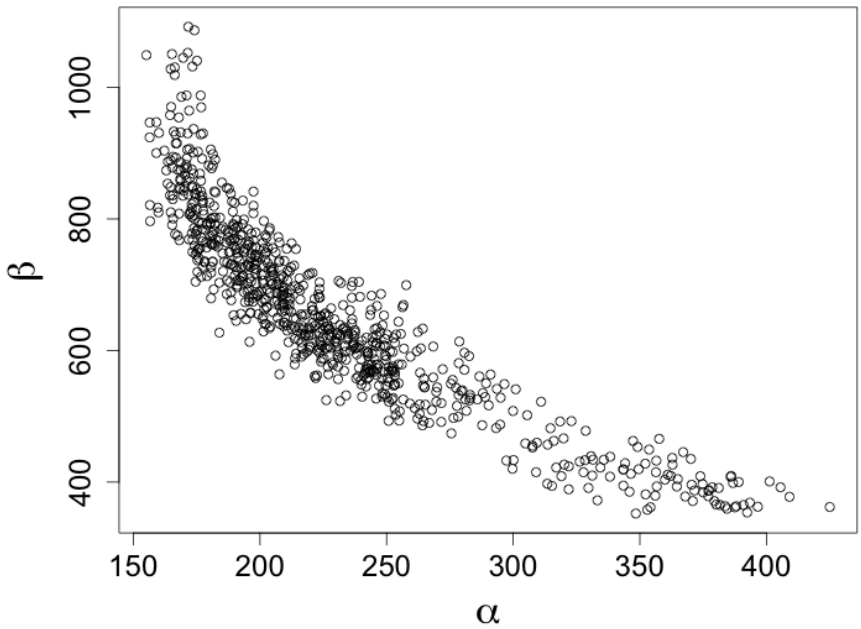

Figure 7.2.: Samples from posterior for α and β, generated by MCMC algorithm.

to get stuck at the boundary of the prior because the MCMC sampler would frequently make

proposals outside of the support of the prior distribution, and these proposals were always rejected.

To understand why this is the case, consider the smMALA algorithm, where local gradient and Hessian information is calculated on each iteration in order to design effective proposals. A hard constraint on the prior density causes the prior density to be discontinuous. The gradient and Hessian will also be discontinuous, which means that local information calculated near the boundary of the prior could lead to very poor parameter proposals. For example a uniform distribution has 0 curvature. In a naive implementation of the smMALA algorithm this lead to proposals with infinite variance. In practice we regularize the Hessian so that the proposal variance is bounded. But if the target distribution has support on $[0, 1]$ and the variance is bounded at 10^6, for example, then (assuming the initial state is inside $[0, 1]$) almost all proposed parameters will be rejected because they will have zero density. It may be possible to resolve this issue by using the variance of the prior to bound the variance of the proposal. We do not further explore that possibility here.

Another issue with applying hard constraints is that the mode of the posterior distribution may lie outside the specified ranges. If this is the case it will cause issues for all types of MCMC algorithm as the MCMC sampler will run up against a boundary and get stuck there. It is then unclear whether the problem is with the constraints, with the model, or possibly with neither. If the parameters of a model are only weakly identifiable, the mode of the posterior distribution can be outside physiological ranges even if the true parameters are within physiological ranges. We found that this was an issue for the Liley *et al* model in that when we simulated data from the model for a given set of parameters, a MwG sampler would get stuck at a boundary. This would happen even when the parameter values at the boundary were not close to the values that were used to simulate the data.

The next thing that I tried was using an unbounded uniform density as the prior. An unbounded uniform density is referred to as an improper prior because it cannot be normalized. If

the data is informative with respect to the parameters this is not necessarily a problem. However, I found that with the Liley *et al* model, using an unbounded uniform prior lead to a posterior with regions of high posterior density for arbitrarily large parameter values. In my MCMC sampler this caused numerical issues as parameters were sampled outside the bound of machine precision. It is also unsatisfactory from a computational point of view to spend a large amount of time exploring regions of the parameter space that are known to be unphysiological.

I then developed a prior that used the ranges to inform the mean and standard deviation of unbounded prior distributions. This approach imposes soft constraints in that parameter values that are far from the expected value will have a low prior density. For example, take the mean to be the mid-point of the interval, and the standard deviation to be one quarter the width of the interval. Then if the prior distribution is normal, approximately 95% of the prior density will be within the physiological ranges.

A possible criticism of this type of approach is that it introduces bias into the parameter estimates because the prior distribution is non-uniform with a mode that is almost surely not located at the true parameters. If parameters are identifiable, this bias decreases as more data is collected. If parameters are not identifiable, the posterior distribution will look identical to the prior distribution. Plotting the prior and the posterior side by side should indicate how much influence the prior distribution has on posterior estimates, at least qualitatively.

7.2.1 *Posterior-prior transformation*

It is often the case that the model parameterization that works best for sampling the posterior is not the same as the parameterization that works best for specifying the prior. A common example of this is when the parameters are log-transformed in the posterior density, which can help to improve sampling efficiency for distributions that are heavy-tailed under the original parameterization and/or are constrained to be positive in the original parameterization. Another example is the parameterization in terms of the equilibrium that was discussed in Chapter 5.

A further example that we identified that is specific to the Liley *et al* model is in the parameterization of the Post-Synaptic Potentials (PSPs). We have found that the total charge transferred tends to be a more informative parameter than the PSP amplitude (Γ_{lk}) or shape parameters, $\gamma_{lk}, \tilde{\gamma}_{lk}$. We use q_{lk} to denote the total charge transferred. The prior information however is specified in terms of PSP amplitudes. These can be calculated in terms of q_{lk}, γ_{lk}, and δ_{lk} (the rise times) as follows,

$$\Gamma_{lk} = \exp(-\gamma_{lk}\,\delta_{lk})\,\gamma_{lk}\,q_{lk}. \tag{7.3}$$

7.3 DEVELOPMENT OF PRIOR DISTRIBUTION FOR EFFECT OF ISOFLURANE

A relatively good knowledge of the relationship between isoflurane concentration and model parameters exists for the Liley *et al* model from independent experiments on rat hippocampal brain slices - see references in [Bojak and Liley, 2005]. These relationships can be quantitatively modelled through Hill equations of the form,

$$H(c) = \frac{K^N + Mc^N}{K^N + c^N}, \tag{7.4}$$

where K, M, and N are the parameters of the Hill equation, and c represents concentration of anaesthetic. The parameter M controls where the minimum of the Hill equation is (for $M < 1$). The parameter K controls the location of the half-maximum, i.e., $H(K) = (1 + M)/2$, which is halfway between 1 and M. The parameter N controls the steepness of the curve with higher values of N resulting in Hill equations with steeper gradients.

In a Bayesian framework we can either incorporate the knowledge of isoflurane effects into the prior distribution (see Sections 7.3.1-7.3.2 for more details) or we can hold it out to use as a validation of the fitting results. Some preliminary experiments that we did (results not shown) indicate that if we do not incorporate the prior beliefs relating to the effect of isoflurane, the parameters are only weakly identifiable. As a result of this we decided to use a more informative prior.

If we want to encode prior knowledge relating to the effect of isoflurane we can either use a functional form, such as a Hill equation, or we can model the difference in parameter values between the different conditions. These effects may be implemented as hard constraints. Under a hard constraint, the value of a parameter in one condition and the isoflurane concentration in both conditions completely determine the parameter value in the other condition. Or the effects may be implemented as soft constraints. Under a soft constraint, the parameter value in the other condition has a probability distribution, so that parameter values that are far away from the expected value of the parameter are penalized through the prior density.

In Section 7.3.1 we use the Hill equations to apply hard constraints to the parameters. These constraints could be developed into soft constraints by (i) treating the coefficients of the Hill equations as unknown parameters in the prior distribution, (ii) allowing for deviations from the Hill function. We do not further consider either of those options here.

In Section 7.3.2 we specify a prior on the ratio of parameters between different conditions. The idea is to introduce some assumptions about what direction the parameters should go in between the different conditions. The constraints are soft in the sense that parameters are allowed to deviate from the expected ratio.

The Hill equations implicitly specify a ratio between parameter values across two conditions. So we expect that if we did implement the Hill equations with soft constraints we would obtain similar results to the ratio prior approach. An advantage of using the Hill equations is that it is possible to predict parameter values at other isoflurane concentrations.

As discussed below, the choice of prior also has computational implications. Under the Hill equation approach, we need to solve a nonlinear (system of) equation(s) for the equilibrium at each concentration. This makes the implementation more challenging, particularly in the case of the Liley *et al* model where the equations for the equilibrium are not always easy to solve. Furthermore, the introduction of a numerical solver makes it more challenging to differentiate the likelihood analytically, although this may be possible with the use of an Automatic Dif-

ferentiation tool. (See Chapter 20 - Solving Algebraic Equations in [Stan Development Team, 2017] or sum of squares minimization in [NAG, 2018] and c05 Chapter introduction of [NAG, 2017]. In contrast the ratio prior is formulated in such a way that the equilibrium can be calculated analytically from the posterior parameters at each isoflurane concentration. This makes implementation easier and saves computational time.

7.3.1 *Hill equations for the effect of isoflurane*

Following [Bojak and Liley, 2005, Bojak et al., 2015] we assume that the amplitude of PSPs decreases with increasing concentration of isoflurane, with the amplitude of excitatory PSPs decaying faster than that of inhibitory PSPs,

$$\Gamma_{ee}(c) = \Gamma^0_{ee} H_e(c), \tag{7.5}$$

$$\Gamma_{ei}(c) = \Gamma^0_{ei} H_e(c), \tag{7.6}$$

$$\Gamma_{ie}(c) = \Gamma^0_{ie} H_i(c), \tag{7.7}$$

$$\Gamma_{ii}(c) = \Gamma^0_{ii} H_i(c), \tag{7.8}$$

where

$$H_e(c) = \frac{0.707^{2.22}}{0.707^{2.22} + c^{2.22}} \tag{7.9}$$

$$H_i(c) = \frac{0.79^{2.6} + 0.56c^{2.6}}{0.79^{2.6} + c^{2.6}}. \tag{7.10}$$

The values of K, M, and N for $H_e(c)$ and $H_i(c)$ have been substituted into the general form given in Equation (7.4).

We also assume that the decay time of inhibitory PSPs increases with increasing concentration of isoflurane, without affecting the PSP rise time. The decay time is completely determined by the parameters γ_{lk} and $\tilde{\gamma}_{lk}$. These parameters are set through a control parameter ϵ_{lk}. The

control parameter ϵ_{lk} is a function of $\kappa_{lk}(c)$ which is the proportional increase in the decay time at isoflurane concentration, c:

$$\gamma_{ie}(c) = \frac{1}{\delta_{ie}} \cdot \frac{\epsilon_{ie}(c)}{\exp[\epsilon_{ie}(c)] - 1} \tag{7.11}$$

$$\gamma_{ii}(c) = \frac{1}{\delta_{ii}} \cdot \frac{\epsilon_{ii}(c)}{\exp[\epsilon_{ii}(c)] - 1} \tag{7.12}$$

$$\tilde{\gamma}_{ie}(c) = \frac{1}{\delta_{ie}} \cdot \exp[\epsilon_{ie}(c)] \tag{7.13}$$

$$\tilde{\gamma}_{ii}(c) = \frac{1}{\delta_{ii}} \cdot \exp[\epsilon_{ii}(c)] \tag{7.14}$$

where

$$\begin{aligned}\epsilon_{ik}(c) = \exp[a_1 - a_2\kappa_{ik}(c)]\sqrt{\kappa_{ik}(c) - 1} + \\ (\exp[a_3(\kappa_{ik}(c) - 1)] - 1) \cdot \left[\frac{1}{\kappa_{ik}^2(c)} + \mathrm{W}_{-1}\left(\frac{\exp[a_4/\kappa_{ik}^2(c)]}{1 - a_5\kappa_{ik}(c)}\right)\right]\end{aligned} \tag{7.15}$$

$$\kappa_{ik}(c) = \frac{0.32^{2.7} + 4.7c^{2.7}}{0.32^{2.7} + c^{2.7}} \tag{7.16}$$

for $k = e$ and $k = i$, with $a_1 = 2.5466$, $a_2 = 1.339$, $a_3 = -1.2699$, $a_4 = -0.2360$, $a_5 = 3.1462$, and W_{-1} the -1 branch of the Lambert-W function.

It can be shown that this choice of $\epsilon_{ik}(c)$ results in decay times that approximately satisfy,

$$\zeta_{ik}(c) = \zeta_{ik}^0 \kappa_{ik}(c). \tag{7.17}$$

The preceding assumptions relating to the effect of isoflurane match the assumptions made in previous studies of the Liley *et al* model [Bojak and Liley, 2005, Bojak et al., 2015]. In addition to this, we introduce a new assumption for the effect of isoflurane on input from the thalamus, based on the results from [Detsch et al., 1999]. We found that without this extra assumption we were not able to obtain good fits to the spectral density in both datasets.

$$\bar{p}_{ee} = \bar{p}_{ee}^0 H_{in}(c) \tag{7.18}$$

$$\sigma_{p_{ee}} = \sigma_{p_{ee}}^0 H_{in}(c) \tag{7.19}$$

$$p_{ei} = p_{ei}^0 H_{in}(c) \tag{7.20}$$

Prior parameters	Generated parameters	Generation method
c_1	-	-
c_2	-	-
Γ^0_{ee}	$\Gamma_{ee}(c_1)$, $\Gamma_{ee}(c_2)$	analytical, Eq. (7.5)
Γ^0_{ei}	$\Gamma_{ei}(c_1)$, $\Gamma_{ei}(c_2)$	analytical, Eq. (7.6)
Γ^0_{ie}	$\Gamma_{ie}(c_1)$, $\Gamma_{ie}(c_2)$	analytical, Eq. (7.7)
Γ^0_{ii}	$\Gamma_{ii}(c_1)$, $\Gamma_{ii}(c_2)$	analytical, Eq. (7.8)
δ_{ie}	$\gamma_{ie}(c_1)$, $\gamma_{ie}(c_2)$, $\tilde{\gamma}_{ie}(c_1)$, $\tilde{\gamma}_{ie}(c_2)$	analytical, Eq. (7.11),(7.13), (7.15)
δ_{ii}	$\gamma_{ii}(c_1)$, $\gamma_{ii}(c_2)$, $\tilde{\gamma}_{ii}(c_1)$, $\tilde{\gamma}_{ii}(c_2)$	analytical, Eq. (7.12),(7.14), (7.15)
$\bar{\sigma}^0_{p_{ee}}$	$\bar{\sigma}_{p_{ee}}(c_1)$, $\bar{\sigma}_{p_{ee}}(c_2)$	analytical, Eq. (7.19)
$\bar{p}^0_{ee}$	$h^*_e(c_1)$, $h^*_e(c_2)$	numerical, Eq. (7.18), (5.13)-(5.14)
$\bar{p}^0_{ei}$	$h^*_i(c_1)$, $h^*_i(c_2)$	numerical, Eq. (7.20), (5.13)-(5.14)

Table 7.2.: Summary of relationship between Hill equation prior and generated parameters used for likelihood evaluation

where

$$H_{in}(c) = \frac{(0.16472^{9.5282} + 0.036595c^{9.5282})}{(0.16472^{9.5282} + c^{9.5282})}. \tag{7.21}$$

7.3.2 *Ratios for the effect of isoflurane*

In the case where we have data from two different conditions we can avoid specifying a functional form for the concentration dependence of parameter values, and simply specify prior beliefs for the difference or ratio between parameter values in the different conditions.

For example we could say that we expect the ratio $\Gamma_{ee}(c_2)/\Gamma_{ee}(c_1)$ to be 0.5 with a standard deviation of 0.1. We refer to this prior as the soft constraints prior, since we do not explore the application of soft constraints using the Hill equations here.

7.4 ANALYSIS OF EEG DATA FROM RATS UNDERGOING ISOFLURANE ANAESTHESIA

In this section we analyze data recorded from rats that have been administered the anaesthetic isoflurane. Concurrent EEG and LFP data were recorded from the barrel cortex of the rat. Data was recorded in resting state and under whisker pad stimulation. Whisker pad stimulation triggers neural activity within the barrel cortex. There is experimental data for three groups of

Prior parameters	Posterior parameters	Posterior to prior map
$\Gamma_{ee}(c_1)$	$q_{ee}(c_1)$	(7.3)
$\Gamma_{ee}(c_2)/\Gamma_{ee}(c_1)$	$q_{ee}(c_2)$	(7.3)
$\Gamma_{ei}(c_1)$	$q_{ei}(c_1)$	(7.3)
$\Gamma_{ei}(c_2)/\Gamma_{ei}(c_1)$	$q_{ei}(c_2)$	(7.3)
$\Gamma_{ie}(c_1)$	$q_{ie}(c_1)$	(7.3)
$\Gamma_{ie}(c_2)/\Gamma_{ie}(c_1)$	$q_{ie}(c_2)$	(7.3)
$\Gamma_{ii}(c_1)$	$q_{ii}(c_1)$	(7.3)
$\Gamma_{ii}(c_2)/\Gamma_{ii}(c_1)$	$q_{ii}(c_2)$	(7.3)
$\epsilon_{ie}(c_1)$	$\epsilon_{ie}(c_1)$	-
$\epsilon_{ie}(c_2)/\epsilon_{ie}(c_1)$	$\epsilon_{ie}(c_2)$	-
$\epsilon_{ii}(c_1)$	$\epsilon_{ii}(c_1)$	-
$\epsilon_{ii}(c_2)/\epsilon_{ii}(c_1)$	$\epsilon_{ii}(c_2)$	-
$\bar{p}_{ee}(c_1)$	$h_e^*(c_1)$	(5.13)-(5.14)
$\bar{p}_{ee}(c_2)/\bar{p}_{ee}(c_2)$	$h_e^*(c_2)$	(5.13)-(5.14)
$p_{ei}(c_1)$	$h_i^*(c_1)$	(5.13)-(5.14)
$p_{ei}(c_2)/p_{ei}(c_2)$	$h_i^*(c_2)$	(5.13)-(5.14)
$\bar{\sigma}_{p_{ee}}(c_1)$	$\bar{\sigma}_{p_{ee}}(c_1)$	-
$\bar{\sigma}_{p_{ee}}(c_2)/\bar{\sigma}_{p_{ee}}(c_1)$	$\bar{\sigma}_{p_{ee}}(c_2)$	-

Table 7.3.: Summary of relationship between soft constraints prior and posterior parameters.

rat - control, old and young. Data was recorded at a frequency of around $24,400$Hz, but this was subsequently downsampled for analysis by a factor of 100 to 244Hz.

Here we analyze data from a single rat (subject 1) in the control group, and consider only the resting state EEG data, at the lightest and deepest anaesthesia levels displayed in Figure 7.3. Following the discussion in Section 3.2 and in common with the existing literature on EEG data analysis using NPMs we assume that the dominant contribution to the EEG signal comes from current dipoles that are generated by excitatory pyramidal neurons in Layer IV / the Granular layer. And that these current dipoles are correlated with the mean membrane potential of the excitatory population.

For model fitting we use the Liley *et al* model equations as outlined in Section 3.1.4, with the biexponential Post Synaptic Potential (PSP) that was discussed in Section 3.3.1. We used the

Whittle likelihood with Fourier frequencies in the range $[0, 20]$Hz. The prior distributions are informed by the ranges in Table 3.2, see Section 7.2 for more details.

When using the Hill equation prior, Table 7.2 lists parameters that need to be generated in order to evaluate the likelihood, with references to the equations used for generation and whether a numerical solver is needed to evaluate the generated parameter values. When using the soft constraints prior, Table 7.3 lists parameters of the prior and of the posterior, and specifies the map that is used to transform from posterior to prior parameters.

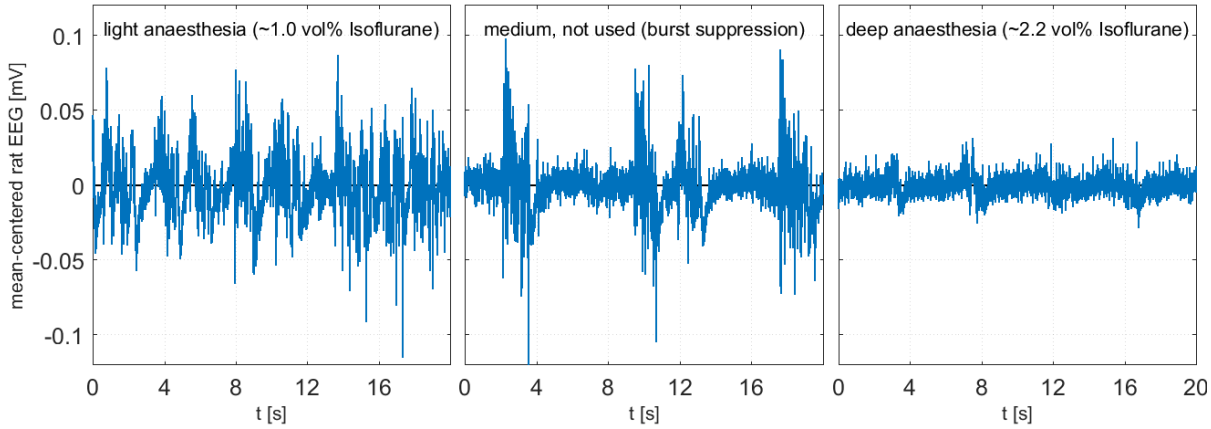

Figure 7.3.: EEG time-series recorded from a single rat at three different level of isoflurane anaesthesia

7.4.1 *Parameter estimation with uncertainty quantification*

Figure 7.4 shows 95% confidence intervals for the spectral density obtained using the Welch method alongside 95% credible intervals for the spectral density estimated from 100,000 MCMC iterations with the Hill equation priors for the isoflurane effect. These samples were obtained by initializing the MCMC sampler at a parameter set identified by Particle Swarm Optimization (PSO) using a separate code in Matlab. I found that I was not able to obtain good fits with my conditional maximization algorithm in C++. This is because the posterior distribution appears to be highly multi-modal. My conditional maximization algorithm is only designed to find local optima whereas PSO is a global optimizer.

There appears to be broad agreement between the Welch estimates and the estimates from the model. The uncertainty around the MCMC estimates of the spectral density is smaller than

it was in the harmonic oscillator example from the previous chapter. This suggests that the MCMC sampler may be mixing slowly and has not fully explored the posterior distribution. It may not even have fully explored a single mode of the posterior distribution. The diagnostics from the MCMC sampler also suggest this may be the case as (i) the variability in the target density is small (`abs(lt_diff)` is mostly in the interval $[0.01, 0.2]$), (ii) the acceptance rate is low (around $10 - 15\%$) even when the step-size parameter in smMALA is relatively small (0.01 compared to 1.0 in the harmonic oscillator). In other words, compared to the sampler for the harmonic oscillator, the proposal distribution for this model is generating smaller moves and accepting them less frequently.

Although the MCMC diagnostics suggest that there is scope for improving the sampling efficiency, we are still able to obtain some quantification of uncertainty in parameter estimates, as shown in Figure 7.5, which would not be possible if only optimization was used. For these experiments the isoflurane concentration is known but I did not use this information in the estimation. The estimates of isoflurane concentration obtained from MCMC appear to be close to the experimental estimates. This suggests that the model is able to reproduce the dynamics of neural activity with physiological parameters, and can recover the underlying concentration of isoflurane from the data.

Figure 7.6 shows the corresponding credible intervals for PSP amplitudes when the soft constraints prior described in Section 7.3.2 is used. These results suggest that the effect of isoflurane on Γ_{ee} differs from the effect of isoflurane on Γ_{ei}, whereas in the Hill equation priors isoflurane has the same effect on Γ_{ee} and Γ_{ei} (in terms of the proportional PSP amplitude decrease per unit increase in concentration). Figure 7.7 shows credible intervals for other concentration dependent parameters in the model. We can see that the effect on isoflurane appears to differ for p_{ee} and p_{ei}. Again this is something that is not possible under the Hill equation prior where Equation (7.21) determines the dependence on isoflurane for p_{ee}, p_{ei}, and $\sigma_{p_{ee}}$.

The log likelihood of the optimal parameters is slightly higher with the soft constraints prior than with the Hill equation prior (results not shown), suggesting that this model provides a better fit to the data. However, model comparison is not trivial because the soft constraints model has more parameters. And visually it looks as though the fit to the data is only marginally better.

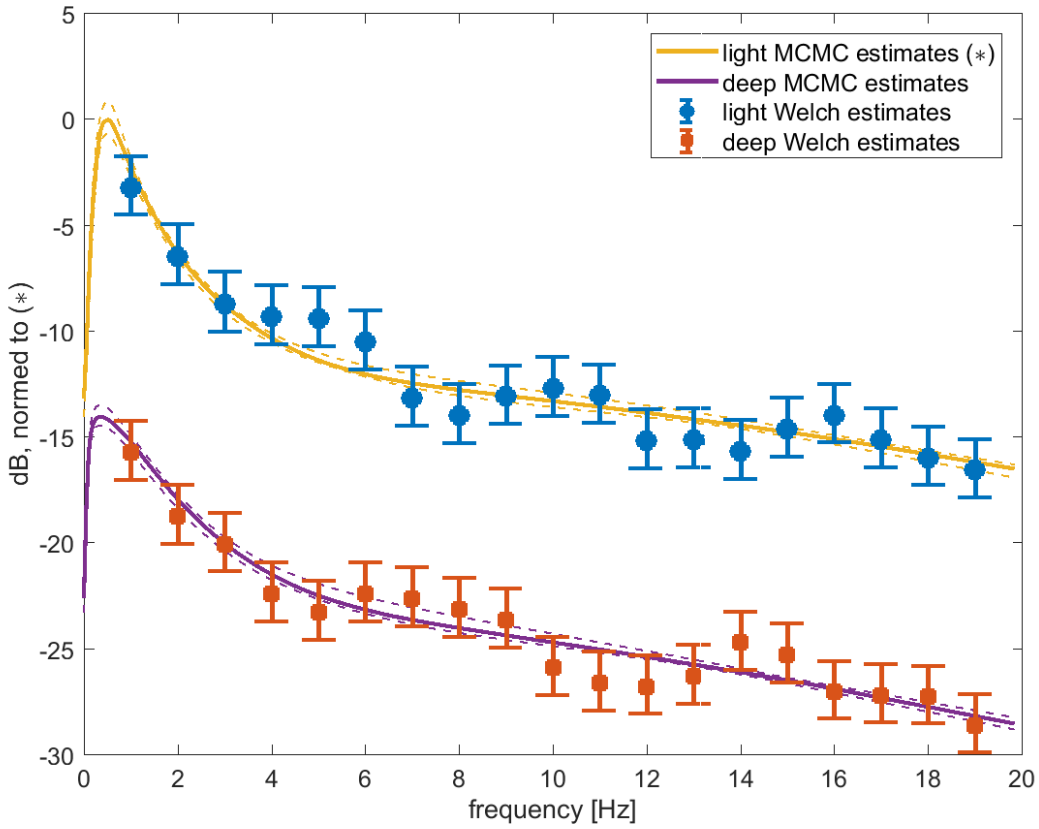

Figure 7.4.: Spectral density estimates for Liley *et al* model from rat data displayed in Figure 7.3. The bars around the Welch estimates are 95% confidence intervals. The dashed lines around the MCMC estimates are 95% credible intervals.

7.5 DISCUSSION

In this chapter we analyzed adult resting state EEG data and then rat anaesthesia EEG data. In our early attempts of fitting the rat data we assumed that isoflurane did not affect the thalamic input parameters in the Liley *et al* model. We found that we were unable to

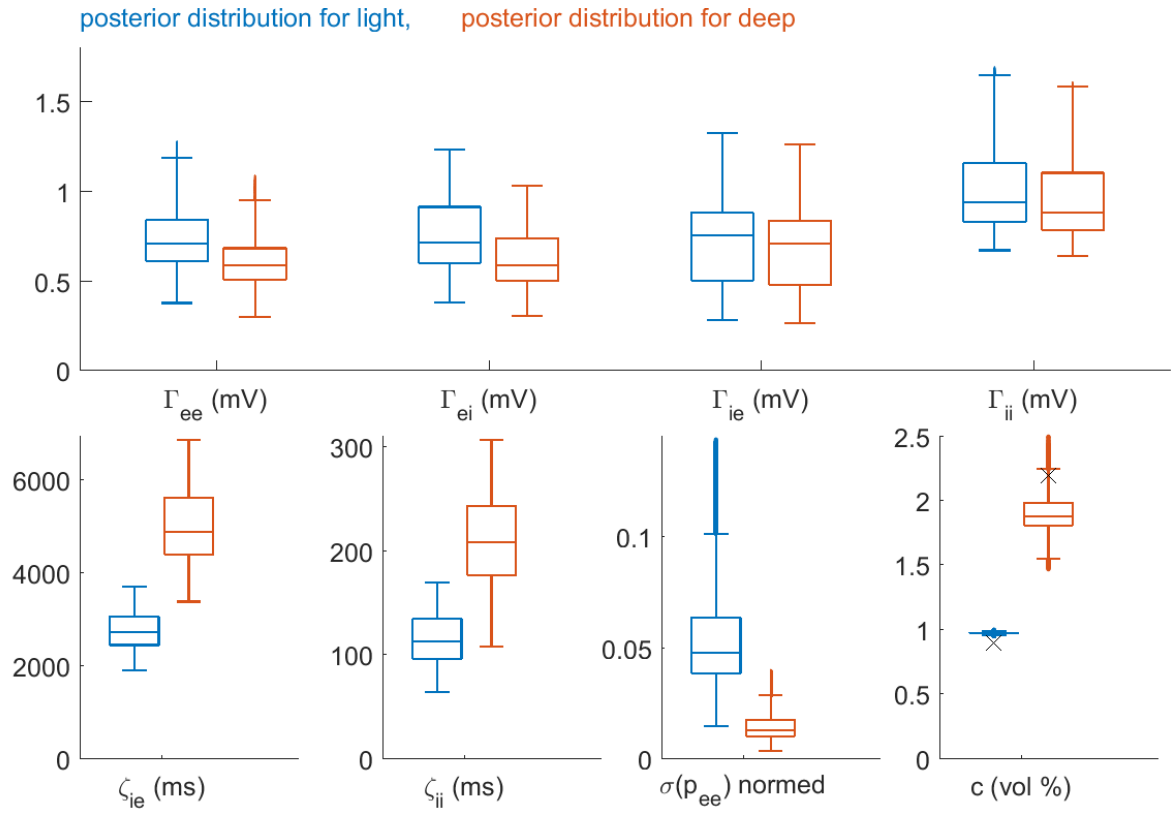

Figure 7.5.: Parameter estimates for Liley *et al* model from rat data displayed in Figure 7.3 with Hill equation prior. Crosses on bottom right figure are independent estimates of isoflurane concentration.

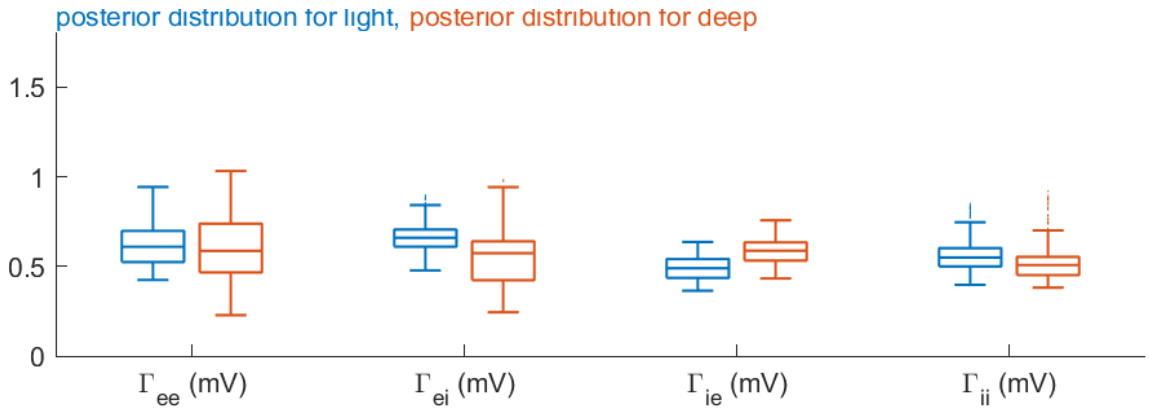

Figure 7.6.: Parameter estimates for PSPs in Liley *et al* model from rat data with soft constraints prior.

obtain good fits with this assumption, which lead us to include these effects, supported by the experimental literature on isoflurane.

When I used the Hill equations to model the effect of isoflurane on model parameters I found that I needed to solve for the steady-state at non-zero concentrations of anaesthetic. This makes it more challenging to calculate analytic derivatives, because the equilibrium is

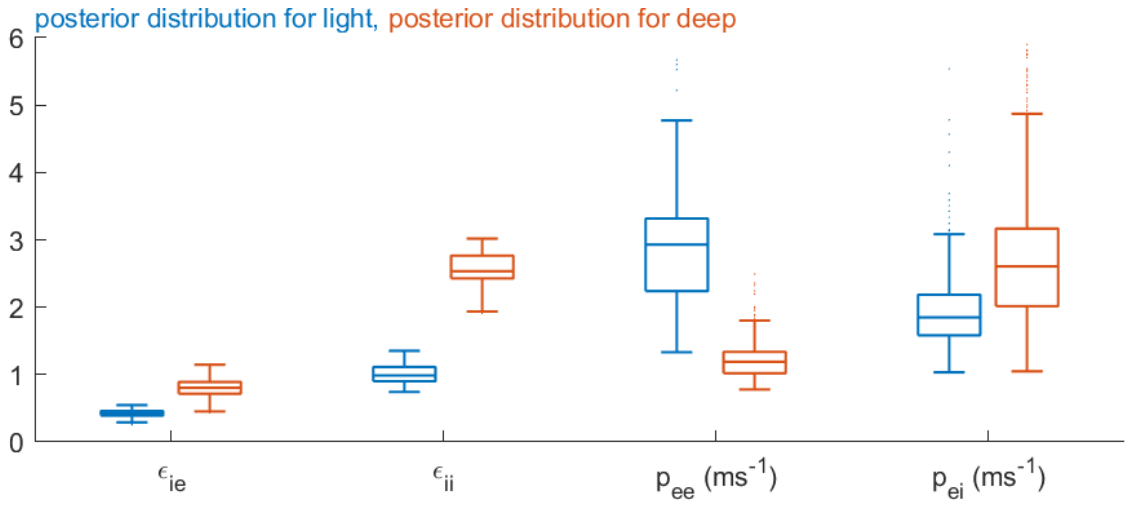

Figure 7.7.: Parameter estimates for other concentration dependent parameters in Liley *et al* model from rat data with soft constraints prior.

no longer an analytical function of the posterior parameters. I also found that obtaining an analytic expression for the Jacobian determinant in the map from posterior to prior parameters was not tractable with the Matlab symbolic toolbox in my computing environment due to the complexity of the symbolic operations. These issues could potentially be addressed through the use of Automatic Differentiation (AD) software. Nevertheless I found that I was able to obtain satisfactory results using a finite difference approximation to the derivatives.

Another area for further work would be to explore the use of SMC algorithms for sampling the posterior distribution. SMC methods can be more effective than MCMC for sampling from multi-modal distributions, but are typically more computationally expensive. So it would be helpful if the code was further optimized (e.g. through the use of analytic derivatives) before this was pursued.

Another reason for using SMC is that it would enable model comparison through the estimation of marginal likelihoods (where both state variables and parameters are integrated out). Some form of model comparison would be helpful for evaluating more quantitatively the credibility of different hypotheses, such as whether or not isoflurane has an effect on thalamic input.

8

SUMMARY

Over the last 70 years mathematical modelling and scientific computing have become cornerstones of science. In some fields, such as numerical weather prediction or financial derivative pricing, mathematical models, such as the Navier-Stokes equations or the Black-Scholes model, are used to make predictions in operational systems. Understanding the human brain - how it gives rise to conscious experience and how it makes intelligent decisions - remains one of the biggest outstanding scientific problems. While there has been some success in developing computational models to further our understanding, most notably the development of the Hodgkin-Huxley equations, most people working in Computational Neuroscience would say that we are a long way from a good understanding of the brain. And therefore a long way from being able to use Computational Neuroscience models to describe and predict brain activity, for example in a clinical setting.

The work in this thesis was motivated by the following problem in Neuroscience: computationally tractable inference of the parameters of Neural Population Models (NPMs) from EEG time-series at different levels of anaesthesia. This is important for understanding how physiological parameters in the brain change during the loss of consciousness.

Due to a lack of knowledge of the parameter values in NPMs, and also the complexity of the models, Bayesian methods for uncertainty quantification are needed to address this problem effectively. In this thesis we have developed computationally tractable methods for Bayesian

inference in stable differential equations. This class of models was described in Chapter 4 and is characterized by approximately linear dynamics around a stable equilibrium. The dynamics of brain activity as observed in resting state EEG data are an example of a phenomenon that can be described by a stable differential equation model and other examples may be found in other areas of science where differential equations models are used. It is worth emphasizing that many differential equation models can have both approximately linear dynamics and fully nonlinear dynamics (such as limit cycles or chaotic attractors) depending on their parameter values. It is often the case that full uncertainty quantification is intractable in the fully nonlinear regime. Our focus has been on the approximately linear regime where uncertainty quantification can still be very challenging.

The state-of-the-art in parameter estimation for noisy dynamical systems is particle MCMC. This method is very generally applicable and produces Monte Carlo estimates that are unbiased. However it scales poorly with the complexity of the model, the number of unknown parameters and the number of observations, making it unsuitable (at the time of writing) for the problems we are interested in. Within Neuroscience the DCM community has developed more scalable methods. In the context of spectral analysis these method linearize NPMs around a stable equilibrium and evaluate likelihoods by comparing the predicted spectral density with an estimate of the spectral density from the data. Whereas particle MCMC is an example of an exact method (as a result of its unbiasedness), spectral DCM is an approximate method in that estimates are biased, and this bias is challenging to quantify.

In Chapters 4 to 5 we focused on the case where we have a single stationary time-series lasting for around 20 seconds, and introduced a number of innovations that enable posterior distributions to be estimated (i) with a computational cost that scales well to relatively complex problems and large volumes of data, (ii) with accuracy that can quantified. In Chapter 4 we analyzed the Whittle likelihood and quantified the errors that the Whittle likelihood introduces into the likelihood evaluation that arise from the asymptotic approximation and from linearizing

the model. We were able to give specific guidance on how long a time-series needs to be in order for the Whittle likelihood to be accurate. And we found that estimates of the posterior distribution under the linearized model tended to over-estimate uncertainty in the sense that some parameter sets that are identified as a good fit by the linearized model actually have a relatively low likelihood. In Chapter 5 we showed how stable differential equation models can be parameterized in terms of their equilibrium values, and demonstrated that this leads to increased sampling efficiency in MCMC.

In Chapters 6 to 7 we extended the approach to the case where we have multiple independent time-series with a view to estimating how model parameters vary with concentration of anaesthesia. In Chapter 6 we looked at a C++ implementation I developed for parameter inference in this setting, using the harmonic oscillator as a simple example of a stable SDE. In Chapter 7 we looked at rat EEG data from two levels of isoflurane anaesthesia and estimated parameters of the Liley *et al* model. This data analysis work led us to include the effects of isoflurane on thalamic input parameters in the Liley *et al* model, whereas most previous work on this model restricted the effect of isoflurane to cortical parameters.

Working from the specific to the general, some possible next steps are as follows. The scalability of my C++ implementation could be further improved. Analytic derivatives were used in my R implementation to generate the results in Chapter 5. Currently my C++ implementation uses finite differences. This is primarily because the derivative calculations are more challenging for the problem in Chapter 7. This issue could be addressed through use of an AD tool such as `dco/c++`. Improving the scalability of the C++ implementation would enable the methods to be applied to larger datasets - currently the MCMC sampling takes several days for analysis of 1-2 minutes of data, but it is not uncommon for EEG recordings to last for hours. A faster C++ implementation would also enable the use of SMC methods, which could lead to more accurate estimation of the posterior. The MCMC method I am currently using is most well suited to unimodal problems. However the results of PSO indicate the presence of multiple modes, so

use of SMC is desirable. Use of SMC would also enable model comparison, for example further exploration of possible isoflurane effects could lead to further refinement of the isoflurane model and a more detailed understanding of anaesthesia.

Next there is a need to persuade other users of stable differential equation models to adopt the methods we have developed. This means that the methods need to be both easy to use and demonstrably better than alternative methods. Compared to the DCM methods, which are part of the SPM package in Matlab, the software I have written is not easy to use. It is therefore important to develop a high-level interface to my C++ code in Python and/or Matlab. SPM users might also question whether the methods I have developed are better than the DCM methods. More empirical comparisons between my methods and DCM methods on a range of different NPMs would help to illuminate this. This would enable better uncertainty quantification in other settings where the use of stable differential equations is appropriate. Potential applications that we are aware of within Neuroscience are (i) quantifying the effect of other anaesthetic agents, such as propofol, and (ii) quantifying the effect of Deep Brain Stimulation in Parkinson's Disease.

Looking beyond the case of approximately linear dynamics, there are several ways in which the work in this thesis could be used or developed. We saw in the data that we analyzed in Chapter 7 that the medium level of anaesthesia was characterized by bursting dynamics. This raises the question of how best to analyze datasets where we wish to use the same model for both approximately linear and fully nonlinear regimes. This is of interest in anaesthesia where models have been developed that can describe light, medium and deep anaesthesia levels. If we are in a setting where we have a large volume of stationary data interspersed with short periods of nonlinear activity, then tractable estimation may be achieved by applying my methods to the stationary data and a particle filter to the nonlinear transients. In order to do this effectively it would be useful to have an efficient particle filter implementation integrated with the C++ code that I have written.

Analysis of large volumes of non-stationary data would likely require the development of novel statistical methods. A significant source of inspiration for our work on using the Whittle likelihood came from the literature on pseudo-marginal algorithms. Whereas the first pseudo-marginal algorithms focused on the case where the likelihood can be estimated unbiasedly, we (and others) have extended the idea to situations where we can evaluate a fast but biased approximation. This work could be further developed by considering other biased likelihoods that apply to a wider range of models (such as the Unscented Kalman Filter and the Ensemble Kalman Filter). And also by using biased estimates within a delayed acceptance framework, which would recover the unbiasedness but still potentially lead to computational savings.

It is possible that in 70 years' time we will have the same level of understanding of the human brain as we currently do of many other physical systems. And that this understanding will be used in a wide range of clinical settings to restore brain function through medical intervention. Getting to that point requires not only advances in the recording of brain activity and the further development of computational models that describe brain dynamics, but also tractable methods for combining computational models with data. The methods that I have developed are scalable and extend the range of Computational Neuroscience models for which parameter estimation with accurate uncertainty quantification is tractable. These methods will inform the next generation of neuroscience data analysis tools and further stimulate the development of statistical methods that can be used to advance scientific research.

A

EIGENVECTOR DERIVATIVES

This appendix describes an alternative method for calculating eigenvector derivatives that preserves the convention that eigenvectors have norm 1. These results are not used in the main thesis as the calculations using eigenvectors that we use do not require the eigenvectors to have a constant norm.

Let A be an $n \times n$ matrix. The matrices R and Λ are an eigen-decomposition of A if,

$$AR = R\Lambda, \tag{A.1}$$

and Λ is a diagonal matrix. The non-zero entries of Λ are referred to as eigenvalues, $\lambda_1, \ldots, \lambda_n$, and the columns of R, are referred to as (right) eigenvectors, $\mathbf{r}_1, \ldots, \mathbf{r}_n$.

By convention, eigenvectors have unit length, i.e.,

$$\|\mathbf{r}_i\| = \mathbf{r}_i^* \cdot \mathbf{r}_i = 1, \quad \text{for all } i, \tag{A.2}$$

where the * operator represents the complex conjugate transpose.

Note that, even with the norming convention, eigenvectors are not unique. If we multiply an eigenvector of A by a unit length complex scalar, it is still an eigenvector of A and it still has the same norm, i.e., if $\mathbf{r}$ is an eigenvector of A, then $\exp(i\beta)\,\mathbf{r}$, with β a real scalar, is also an eigenvector A.

We can uniquely define an eigenvector by adding the condition,

$$\mathbf{r}(k) = |\mathbf{r}(k)|, \tag{A.3}$$

i.e. the kth entry of $\mathbf{r}$ should be equal to its absolute value. This condition can be satisfied by multiplying the original eigenvector by a complex scalar, $\alpha = |\mathbf{r}(k)|/\mathbf{r}(k)$. This scalar has unit length, so $\alpha\,\mathbf{r}$ is an eigenvector of A, and has the same norm as $\mathbf{r}$. The index k can be chosen as the index of any non-zero component in $\mathbf{r}$.

Suppose we would like to analytically evaluate the derivative of an eigenvalue,

$$\lambda_\theta = \frac{d\lambda}{d\theta}, \tag{A.4}$$

and its corresponding eigenvector,

$$\mathbf{r}_\theta = \left(\frac{dr_1}{d\theta}, \ldots, \frac{dr_1}{d\theta}\right), \tag{A.5}$$

where the subscript now denotes elementwise differentiation with respect to a scalar parameter θ.

We have,

$$(A\mathbf{r})_\theta = (\lambda\mathbf{r})_\theta \tag{A.6}$$

$$\implies (A - \lambda I)\,\mathbf{r}_\theta - \mathbf{r}\lambda_\theta = -A_\theta\mathbf{r}. \tag{A.7}$$

Assuming that we have already found $\mathbf{r}$ and λ, and that we can differentiate each element of A with respect to θ, Equation (A.7) is a system of n linear equations with $n+1$ unknown complex numbers.

The equation can be re-written as a system of $2n$ linear equations in $2n+2$ unknown real numbers by taking the real and imaginary parts of each side of Equation (A.7).

Assuming, as before, that the imaginary part of A is zero, and defining $\mathbf{r} = \mathbf{x} + i\mathbf{y}$, and $\lambda = \lambda^x + i\lambda^y$, we have,

$$\mathrm{Re}[(A-\lambda I)\mathbf{r}_\theta] = A\mathbf{x}_\theta - \lambda^x\mathbf{x}_\theta + \lambda^y\mathbf{y}_\theta \quad \mathrm{Im}[(A-\lambda I)\mathbf{r}_\theta] = A\mathbf{y}_\theta - \lambda^y\mathbf{x}_\theta - \lambda^x\mathbf{y}_\theta$$

$$\mathrm{Re}(\lambda_\theta\mathbf{r}) = \lambda^x_\theta\mathbf{x} - \lambda^y_\theta\mathbf{y} \qquad \mathrm{Im}(\lambda_\theta\mathbf{r}) = \lambda^y_\theta\mathbf{x} + \lambda^x_\theta\mathbf{y}.$$

$$\mathrm{Re}(A_\theta\mathbf{r}) = A_\theta\mathbf{x} \qquad \mathrm{Im}(A_\theta\mathbf{r}) = A_\theta\mathbf{y}$$

Equation (A.7) is therefore equivalent to,

$$\left[\begin{array}{c|c|c|c} A-\lambda^x I & \lambda^y I & \mathbf{x} & -\mathbf{y} \\ \hline -\lambda^y I & A-\lambda^x I & \mathbf{y} & \mathbf{x} \end{array}\right] \left[\begin{array}{c} \mathbf{x}_\theta \\ \hline \mathbf{y}_\theta \\ \hline \lambda^x_\theta \\ \hline \lambda^y_\theta \end{array}\right] = \left[\begin{array}{c} -A_\theta\,\mathbf{x} \\ \hline -A_\theta\,\mathbf{y} \end{array}\right]. \tag{A.8}$$

The first n equations correspond to the real parts of Equation (A.7) and equations $n+1$ to $2n$ correspond to the imaginary parts.

In order to fully determine the system we need to differentiate the constraints that we defined in the previous section that uniquely define the eigenvector, $\mathbf{r}$.

$$\mathbf{r}^* \cdot \mathbf{r} = 1 \tag{A.9}$$

$$\implies \quad \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} = 1 \tag{A.10}$$

$$\implies \quad \mathbf{x}^T\mathbf{x}_\theta + \mathbf{y}^T\mathbf{y}_\theta = 0 \tag{A.11}$$

And,

$$\mathbf{r}(k) = |\mathbf{r}(k)| \tag{A.12}$$

$$\implies \quad \mathbf{y}(k) = 0 \tag{A.13}$$

$$\implies \quad \mathbf{y}_\theta(k) = 0 \tag{A.14}$$

$$\tag{A.15}$$

The full linear system for the eigenvector and eigenvalue derivatives can then be written as,

$$\left[\begin{array}{c|c|c|c} A - \lambda^x I & \lambda^y I & \mathbf{x} & -\mathbf{y} \\ \hline -\lambda^y I & A - \lambda^x I & \mathbf{y} & \mathbf{x} \\ \hline \mathbf{x}^T & \mathbf{y}^T & 0 & 0 \\ \hline 0 & \mathbf{i}_k^T & 0 & 0 \end{array}\right] \left[\begin{array}{c} \mathbf{x}_\theta \\ \hline \mathbf{y}_\theta \\ \hline \lambda_\theta^x \\ \hline \lambda_\theta^y \end{array}\right] = \left[\begin{array}{c} -A_\theta\ \mathbf{x} \\ \hline -A_\theta\ \mathbf{y} \\ \hline 0 \\ \hline 0 \end{array}\right]. \tag{A.16}$$

where $\mathbf{i}_k$ is a vector with a one in element k and zeroes elsewhere.

BIBLIOGRAPHY

[Abeysuriya and Robinson, 2016] Abeysuriya, R. and Robinson, P. (2016). Real-time automated EEG tracking of brain states using neural field theory. *Journal of Neuroscience Methods*, 258:28–45.

[Alquier et al., 2016] Alquier, P., Friel, N., Everitt, R., and Boland, A. (2016). Noisy Monte Carlo: Convergence of Markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47.

[Anderson and Moore, 1979] Anderson, B. D. and Moore, J. B. (1979). Optimal filtering. *Prentice-Hall*, 21:22–95.

[Andrieu et al., 2010] 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.

[Andrieu and Roberts, 2009] Andrieu, C. and Roberts, G. O. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725.

[Andrieu and Thoms, 2008] Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statistics and Computing*, 18(4):343–373.

[Andrieu and Vihola, 2015] Andrieu, C. and Vihola, M. (2015). Convergence properties of pseudo-marginal Markov Chain Monte Carlo algorithms. *The Annals of Applied Probability*, 25(2):1030–1077.

[Andrzejak et al., 2001] Andrzejak, R. G., Lehnertz, K., Mormann, F., Rieke, C., David, P., and Elger, C. E. (2001). Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907.

[Beaumont, 2003] Beaumont, M. A. (2003). Estimation of population growth or decline in genetically monitored populations. *Genetics*, 164(3):1139–1160.

[Beskos et al., 2014] Beskos, A., Crisan, D., Jasra, A., Kamatani, K., and Zhou, Y. (2014). A stable particle filter in high-dimensions. *arXiv preprint arXiv:1412.3501*.

[Betancourt, 2013] Betancourt, M. (2013). A general metric for Riemannian manifold Hamiltonian Monte Carlo. In *Geometric science of information*, pages 327–334. Springer.

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

[Blei et al., 2017] 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.

[Bojak and Liley, 2005] Bojak, I. and Liley, D. (2005). Modeling the effects of anesthesia on the electroencephalogram. *Physical Review E*, 71(4):041902.

[Bojak and Liley, 2010] Bojak, I. and Liley, D. T. (2010). Axonal velocity distributions in neural field equations. *PLoS Computational Biology*, 6(1):e1000653.

[Bojak et al., 2015] Bojak, I., Stoyanov, Z. V., and Liley, D. T. (2015). Emergence of spatially heterogeneous burst suppression in a neural field model of electrocortical activity. *Frontiers in Systems Neuroscience*, 9:18.

[Breakspear and Terry, 2002] Breakspear, M. and Terry, J. (2002). Detection and description of non-linear interdependence in normal multichannel human EEG data. *Clinical Neurophysiology*, 113(5):735–753.

[Bressloff, 2011] Bressloff, P. C. (2011). Spatiotemporal dynamics of continuum neural fields. *Journal of Physics A: Mathematical and Theoretical*, 45(3):033001.

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

[Calderhead and Girolami, 2011] Calderhead, B. and Girolami, M. (2011). Statistical analysis of nonlinear dynamical systems using differential geometric sampling methods. *Interface focus*, 1(6).

[Carpenter et al., 2017] Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1).

[Carpenter et al., 2015] Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., and Betancourt, M. (2015). The Stan math library: Reverse-mode automatic differentiation in C++. *arXiv preprint arXiv:1509.07164*.

[Chopin and Ridgway, 2017] Chopin, N. and Ridgway, J. (2017). Leave Pima Indians alone: binary regression as a benchmark for Bayesian computation. *Statistical Science*, 32(1):64–87.

[Contreras-Cristán et al., 2006] Contreras-Cristán, A., Gutiérrez-Peña, E., and Walker, S. G. (2006). A note on Whittle's likelihood. *Communications in Statistics - Simulation and Computation*, 35(4):857–875.

[Dahlin et al., 2015] Dahlin, J., Lindsten, F., and Schön, T. B. (2015). Particle Metropolis–Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92.

[David and Friston, 2003] David, O. and Friston, K. J. (2003). A neural mass model for MEG/EEG:: coupling and neuronal dynamics. *NeuroImage*, 20(3):1743–1755.

[David et al., 2005] David, O., Harrison, L., and Friston, K. J. (2005). Modelling event-related responses in the brain. *NeuroImage*, 25(3):756–770.

[Dayan and Abbott, 2001] Dayan, P. and Abbott, L. F. (2001). *Theoretical neuroscience*. Cambridge, MA: MIT Press.

[Deco et al., 2008] Deco, G., Jirsa, V. K., Robinson, P. A., Breakspear, M., and Friston, K. (2008). The dynamic brain: from spiking neurons to neural masses and cortical fields. *PLoS Computational Biology*, 4(8):e1000092.

[Del Moral et al., 2006] Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

[Detsch et al., 1999] Detsch, O., Vahle-Hinz, C., Kochs, E., Siemers, M., and Bromm, B. (1999). Isoflurane induces dose-dependent changes of thalamic somatosensory information transfer. *Brain Research*, 829(1-2):77–89.

[Doucet et al., 2001] Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer.

[Doucet and Johansen, 2009] Doucet, A. and Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3.

[Doucet et al., 2015] Doucet, A., Pitt, M., Deligiannidis, G., and Kohn, R. (2015). Efficient implementation of Markov Chain Monte Carlo when using an unbiased likelihood estimator. *Biometrika*, 102(2):295–313.

[Eilers and Goeman, 2004] Eilers, P. H. and Goeman, J. J. (2004). Enhancing scatterplots with smoothed densities. *Bioinformatics*, 20(5):623–628.

[Elvira et al., 2018] Elvira, V., Martino, L., and Robert, C. P. (2018). Rethinking the effective sample size. *arXiv preprint arXiv:1809.04129*.

[Evensen, 1994] Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162.

[Everitt, 2012] Everitt, R. G. (2012). Bayesian parameter estimation for latent markov random fields and social networks. *Journal of Computational and Graphical Statistics*, 21(4):940–960.

[Friston et al., 2007] Friston, K., Mattout, J., Trujillo-Barreto, N., Ashburner, J., and Penny, W. (2007). Variational free energy and the Laplace approximation. *NeuroImage*, 34(1):220–234.

[Gelman et al., 1996] Gelman, A., Bois, F., and Jiang, J. (1996). Physiological pharmacokinetic analysis using population modeling and informative prior distributions. *Journal of the American Statistical Association*, 91(436):1400–1412.

[Gelman et al., 2014] Gelman, A., Stern, H. S., Carlin, J. B., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian data analysis: Third Edition*. Chapman and Hall/CRC.

[Giles, 2008] Giles, M. (2008). An extended collection of matrix derivative results for forward and reverse mode automatic differentiation. Technical report, Oxford.

[Gilks et al., 1995] Gilks, W. R., Richardson, S., and Spiegelhalter, D. (1995). *Markov Chain Monte Carlo in practice*. CRC press.

[Girolami and Calderhead, 2011] Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214.

[Golightly et al., 2015] 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.

[Gordon et al., 1993] 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*, volume 140, pages 107–113. IET.

[Green et al., 2015] Green, P. J., Łatuszyński, K., Pereyra, M., and Robert, C. P. (2015). Bayesian computation: a summary of the current state, and samples backwards and forwards. *Statistics and Computing*, 25(4):835–862.

[Grewal and Andrews, 1993] Grewal, M. S. and Andrews, A. P. (1993). *Kalman filtering: Theory and Practice*. Prentice-Hall.

[Griffith, 1963] Griffith, J. (1963). A field theory of neural nets: I: Derivation of field equations. *The Bulletin of Mathematical Biophysics*, 25(1):111–120.

[Hashemi et al., 2018] Hashemi, M., Hutt, A., Buhry, L., and Sleigh, J. (2018). Optimal model parameter estimation from EEG power spectrum features observed during general anesthesia. *Neuroinformatics*, 16(2):231–251.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning*. Springer.

[Hodgkin and Huxley, 1952] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544.

[Hoffman and Gelman, 2014] Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn Sampler: adaptively setting path lengths in hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

[Hutt and Buhry, 2014] Hutt, A. and Buhry, L. (2014). Study of gabaergic extra-synaptic tonic inhibition in single neurons and neural populations by traversing neural scales: application to propofol-induced anaesthesia. *Journal of Computational Neuroscience*, 37(3):417–437.

[Hutt and Longtin, 2010] Hutt, A. and Longtin, A. (2010). Effects of the anesthetic agent propofol on neural populations. *Cognitive Neurodynamics*, 4(1):37–59.

[Izhikevich, 2007] Izhikevich, E. M. (2007). *Dynamical systems in neuroscience*. MIT press.

[Jasra et al., 2007] Jasra, A., Stephens, D. A., and Holmes, C. C. (2007). On population-based simulation for static inference. *Statistics and Computing*, 17(3):263–279.

[Jirsa and Haken, 1996] Jirsa, V. K. and Haken, H. (1996). Field theory of electromagnetic brain activity. *Physical Review Letters*, 77(5):960.

[John et al., 2001] John, E., Prichep, L., Kox, W., Valdes-Sosa, P., Bosch-Bayard, J., Aubert, E., Tom, M., and Gugino, L. (2001). Invariant reversible QEEG effects of anesthetics. *Consciousness and Cognition*, 10(2):165–183.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–194. International Society for Optics and Photonics.

[Kalat, 2015] Kalat, J. W. (2015). *Biological psychology*. Nelson Education.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[Kantas et al., 2014] Kantas, N., Beskos, A., and Jasra, A. (2014). Sequential monte carlo methods for high-dimensional inverse problems: A case study for the Navier–Stokes equations. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):464–489.

[Kantas et al., 2015] Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., and Chopin, N. (2015). On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351.

[Karagiannis and Andrieu, 2013] Karagiannis, G. and Andrieu, C. (2013). Annealed importance sampling reversible jump MCMC algorithms. *Journal of Computational and Graphical Statistics*, 22(3):623–648.

[Kelley, 1967] Kelley, A. (1967). The stable, center-stable, center, center-unstable, unstable manifolds. *Journal of Differential Equations*, 3(4):546–570.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE.

[Kuizenga et al., 2001] Kuizenga, K., Wierda, J., and Kalkman, C. (2001). Biphasic EEG changes in relation to loss of consciousness during induction with thiopental, propofol, etomidate, midazolam or sevoflurane. *British Journal of Anaesthesia*, 86(3):354–360.

[Liepe et al., 2014] 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.

[Liley et al., 2002] Liley, D. T., Cadusch, P. J., and Dafilis, M. P. (2002). A spatially continuous mean field theory of electrocortical activity. *Network: Computation in Neural Systems*, 13(1):67–113.

[Lindén et al., 2011] Lindén, H., Tetzlaff, T., Potjans, T. C., Pettersen, K. H., Grün, S., Diesmann, M., and Einevoll, G. T. (2011). Modeling the spatial reach of the LFP. *Neuron*, 72(5):859–872.

[Lindgren et al., 2011] Lindgren, F., Rue, H., and Lindström, J. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498.

[Lippman et al., 2012] Lippman, S. B., Lajoie, J., and Moo, B. E. (2012). *C++ Primer, 5th edition*. Addison Wesley.

[Lueckmann et al., 2017] Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems*, pages 1289–1299.

[Marreiros et al., 2013] Marreiros, A. C., Cagnan, H., Moran, R. J., Friston, K. J., and Brown, P. (2013). Basal ganglia–cortical interactions in Parkinsonian patients. *Neuroimage*, 66:301–310.

[Maskell, 2004] Maskell, S. R. (2004). *Sequentially structured Bayesian solutions*. PhD thesis, University of Cambridge.

[Mezura-Montes and Coello, 2011] Mezura-Montes, E. and Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm and Evolutionary Computation*, 1(4):173–194.

[Moran et al., 2009] Moran, R. J., Stephan, K. E., Seidenbecher, T., Pape, H.-C., Dolan, R. J., and Friston, K. J. (2009). Dynamic causal models of steady-state responses. *NeuroImage*, 44(3):796–811.

[Murray, 2013] Murray, L. M. (2013). Bayesian state-space modelling on high-performance hardware using LibBi. *arXiv preprint arXiv:1306.3277*.

[NAG, 2017] NAG (2017). *NAG C Library*. The Numerical Algorithms Group Ltd. Mark 26, available at `https://www.nag.com/content/software-documentation`.

[NAG, 2018] NAG (2018). *NAG AD Library*. The Numerical Algorithms Group Ltd. Mark 26, available at `https://www.nag.com/numeric/cl/nagdoc_fl26.2/adhtml/index.html`.

[Neal, 2001] Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

[Neal, 2011] Neal, R. M. (2011). MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman & Hall.

[Nemeth et al., 2016a] Nemeth, C., Fearnhead, P., and Mihaylova, L. (2016a). Particle approximations of the score and observed information matrix for parameter estimation in state–space models with linear computational cost. *Journal of Computational and Graphical Statistics*, 25(4):1138–1157.

[Nemeth et al., 2016b] Nemeth, C., Sherlock, C., and Fearnhead, P. (2016b). Particle Metropolis-adjusted Langevin algorithms. *Biometrika*, 103(3):701–717.

[Nunez and Srinivasan, 1981] Nunez, P. and Srinivasan, R. (1981). *Electric fields of the brain: The neurophysiology of EEG*. New York: Oxford University Press.

[Pedersen, 2010] Pedersen, M. E. H. (2010). Good parameters for particle swarm optimization. *Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001*.

[Penny and Sengupta, 2016] Penny, W. and Sengupta, B. (2016). Annealed importance sampling for neural mass models. *PLoS Computational Biology*, 12(3):e1004797.

[Penny et al., 2011] Penny, W. D., Friston, K. J., Ashburner, J. T., Kiebel, S. J., and Nichols, T. E. (2011). *Statistical parametric mapping: the analysis of functional brain images*. Elsevier.

[Pesaran et al., 2018] Pesaran, B., Vinck, M., Einevoll, G. T., Sirota, A., Fries, P., Siegel, M., Truccolo, W., Schroeder, C. E., and Srinivasan, R. (2018). Investigating large-scale brain dynamics using field potential recordings: analysis and interpretation. *Nature Neuroscience*.

[Poyiadjis et al., 2011] Poyiadjis, G., Doucet, A., and Singh, S. S. (2011). Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80.

[Priestley, 1981] Priestley, M. (1981). *Spectral analysis and time series*. Academic press.

[Raferty and Lewis, 1996] Raferty, A. and Lewis, S. (1996). The number of iterations, convergence diagnostics and generic Metropolis algorithms. In *Markov Chain Monte Carlo in Practice*. Chapman & Hall.

[Rampil, 1998] Rampil, I. J. (1998). A primer for EEG signal processing in anesthesia. *Anesthesiology: The Journal of the American Society of Anesthesiologists*, 89(4):980–1002.

[Rebeschini and Van Handel, 2015] Rebeschini, P. and Van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866.

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

[Robinson et al., 2001] Robinson, P., Rennie, C., Wright, J., Bahramali, H., Gordon, E., and Rowe, D. (2001). Prediction of electroencephalographic spectra from neurophysiology. *Physical Review E*, 63(2):021903.

[Rue et al., 2017] Rue, H., Riebler, A., Sørbye, S. H., Illian, J. B., Simpson, D. P., and Lindgren, F. K. (2017). Bayesian computing with INLA: a review. *Annual Review of Statistics and its Application*, 4:395–421.

[Shumway and Stoffer, 2011] Shumway, R. H. and Stoffer, D. S. (2011). *Time series analysis and its applications*. Springer.

[Stan Development Team, 2017] Stan Development Team (2017). *Stan Modeling Language: User's Guide and Reference Manual*. Stan Version 2.17.0, downloaded from `http://mc-stan.org/users/documentation/`.

[Sykulski et al., 2016] Sykulski, A. M., Olhede, S. C., and Lilly, J. M. (2016). The de-biased Whittle likelihood for second-order stationary stochastic processes. *arXiv*.

[Van Der Merwe et al., 2001] Van Der Merwe, R., Doucet, A., De Freitas, N., and Wan, E. A. (2001). The unscented particle filter. In *Advances in Neural Information Processing Systems*. MIT Press.

[Van Loan, 1978] Van Loan, C. (1978). Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404.

[van Rotterdam et al., 1982] van Rotterdam, A., Da Silva, F. L., Van den Ende, J., Viergever, M., and Hermans, A. (1982). A model of the spatial-temporal characteristics of the alpha rhythm. *Bulletin of Mathematical Biology*, 44(2):283–305.

[Weisstein, 2018] Weisstein, E. W. (2018). Wiener process. From MathWorld—A Wolfram Web Resource. Last visited on 18/5/2018.

[Wheeler et al., 2014] Wheeler, M. W., Dunson, D. B., Pandalai, S. P., Baker, B. A., and Herring, A. H. (2014). Mechanistic hierarchical gaussian processes. *Journal of the American Statistical Association*, 109(507):894–904.

[Williamson et al., 2013] Williamson, D., Goldstein, M., Allison, L., Blaker, A., Challenor, P., Jackson, L., and Yamazaki, K. (2013). History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate Dynamics*, 41(7-8):1703–1729.

[Zhou et al., 2016] Zhou, Y., Johansen, A. M., and Aston, J. A. (2016). Toward automatic model comparison: an adaptive sequential monte carlo approach. *Journal of Computational and Graphical Statistics*, 25(3):701–726.