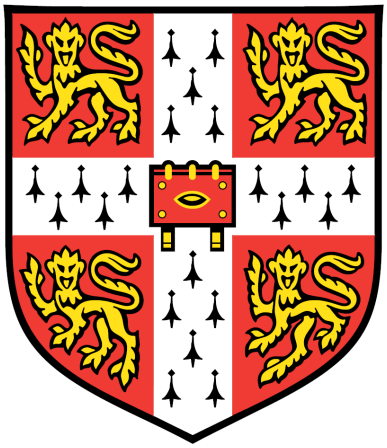
Hidden States, Hidden Structures: Bayesian Learning in Time Series Models

James Murphy

Darwin College
University of Cambridge



This dissertation is submitted for the degree of
Doctor of Philosophy to the University of Cambridge

November 2013

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. No part of this dissertation has been submitted for any other qualification.

James Murphy

Publications

Work from this thesis has been accepted for publication in the following journal and conference articles.

[1] J. Murphy and S. Godsill, "Joint Bayesian removal of impulse and background noise," in Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 261–264, IEEE, 2011.

[2] H. Christensen, J. Murphy[1], and S. Godsill, "Forecasting high-frequency futures returns using online Langevin dynamics," IEEE Journal of Selected Topics in Signal Processing, vol. 6, no. 4, pp. 366–380, 2012.

[3] J. Murphy and S. Godsill, "Simultaneous localization and mapping for non-parametric potential field environments," in Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF2012), pp. 1–6, IEEE, 2012.

[4] J. Murphy and S. Godsill, "Structure inference for networks with general non-parametric inter-object relationships," in Proceedings of 15th International Conference on Information Fusion, IEEE, July 2012.

[1] joint first author

Abstract

This thesis presents methods for the inference of system state and the learning of model structure for a number of hidden-state time series models, within a Bayesian probabilistic framework. Motivating examples are taken from application areas including finance, physical object tracking and audio restoration. The work in this thesis can be broadly divided into three themes: system and parameter estimation in linear jump-diffusion systems, non-parametric model (system) estimation and batch audio restoration.

For linear jump-diffusion systems, efficient state estimation methods based on the variable rate particle filter are presented for the general linear case (chapter 3) and a new method of parameter estimation based on Particle MCMC methods is introduced and tested against an alternative method using reversible-jump MCMC (chapter 4).

Non-parametric model estimation is examined in two settings: the estimation of non-parametric environment models in a SLAM-style problem, and the estimation of the network structure and forms of linkage between multiple objects. In the former case, a non-parametric Gaussian process prior model is used to learn a potential field model of the environment in which a target moves. Efficient solution methods based on Rao-Blackwellized particle filters are given (chapter 5). In the latter case, a new way of learning non-linear inter-object relationships in multi-object systems is developed, allowing complicated inter-object dynamics to be learnt and causality between objects to be inferred. Again based on Gaussian process prior assumptions, the method allows the identification of a wide range of relationships

between objects with minimal assumptions and admits efficient solution, albeit in batch form at present (chapter 6).

Finally, the thesis presents some new results in the restoration of audio signals, in particular the removal of impulse noise (pops and clicks) from audio recordings (chapter 7).

Acknowledgements

There are a great many people without whom this thesis and research would not have been possible.

Firstly, I would like to thank my supervisor, Prof. Simon Godsill, for his guidance, his insightful ideas, helpful comments and for his reassurance from time to time that things were not going too badly. Without his advice this work would certainly not have been possible. It has been a pleasure to work with him and I have greatly enjoyed the PhD process under his guidance.

I am grateful to the EPSRC for providing funding, and to Cantab Capital Partners for further funding during the writing of this thesis.

I would like to thank the SigProC group at the Cambridge University Engineering Department (CUED) for hosting me as a visitor throughout my PhD. Thank-you for welcoming me into a friendly and inclusive research group. Many helpful discussions, interesting ideas and entertaining lunchtimes have stemmed from the coffee room. In particular I would like to thank my fellow PhD students Jens and Tim for their openness, helpfulness and for going through this experience alongside me.

Many people in both the Mathematics and Engineering departments have helped with the sometimes complicated administrative aspects of my PhD. My advisor Prof. Chris Rogers has always been ready to give support and guidance, and has helped me throughout the PhD.

Finally, I would like to thank my family. My parents, Veronica and Kevin, have given me unwavering support both through the PhD process

and more generally throughout my life. Thank-you for always being there for me, for the many things you have taught me and for the love I have always felt from you.

My fiancée Kate has given me limitless support and encouragement throughout. She has put up with a lot. Without her tolerance, humour and love this would have been a much more difficult and lonely experience.

Thank-you all.

Contents

CONTENTS

Chapter 1

Introduction

Time series arise in any situation in which data are collected periodically. They are common throughout science, technology and the humanities. As both sensors and computing power become cheaper more ubiquitous there are ever more opportunities to gather and analyse such data. Many methods have been developed to do so, but amongst the most successful techniques of recent years have been Bayesian statistical approaches. These are based on the idea of quantifying uncertainty in knowledge about the system as probability distributions over the possible states of the properties of interest. Since observations are imperfect, uncertainty about the system is always present in estimation; Bayesian methods aim to formalize and quantify this in a coherent and rational way.

Several justifications exist for Bayesian inference, reflecting different ways of viewing the meaning of uncertainty. The *objective* Bayesian view is that uncertainty about the plausibility of a statement measured by probability is something that, ideally, should be universally agreed upon, given the same data. Thus, the aim of objective Bayesians is to create methods that require minimally subjective components such as priors. In this way these methods try to instill Bayesian reasoning with a validity that can be universally accepted [5]. This is especially appealing when trying to communicate the findings of a Bayesian analysis.

On the other hand, *subjective* Bayesians see probability as a measure of an individual's beliefs about the plausibility of statements. They argue that this is a more honest view because, in practice, objective Bayesian analysis cannot really give true objectivity, but rather only a more objective form of subjectivity. Bruno de Finetti, a key advocate of the subjective interpretation of probability, stated that there was no such thing as probability [6], by which he meant to reject all interpretations of probability as anything other than a statement of subjective belief. He went so far as to liken the notion of the objective existence of probability as a measure of plausibility to a misleading superstition similar to a belief in witches and fairies. An obvious criticism of subjectivist analysis is to question whether conclusions drawn can be meaningful to anyone other than the individual drawing them. A counter-argument is that all methods are subjective in some way and that the subjective approach, by clearly articulating the assumptions and process upon which an analysis is based allows outside observers to decide for themselves whether to reject or accept the findings based on their own subjective assumptions [7]. A sufficiently strong case presented in such a way should be widely convincing without making potentially misleading claims to objectivity.

In either case (and this thesis leans more closely to the subjective view), it is necessary to establish that probability and Bayesian inference are rational and coherent methods for dealing with *certain types* of uncertainty, specifically, uncertainty about things that can be known (so-called epistemic and aleatory uncertainty). Several approaches have been proposed for this, with perhaps the most influential being Cox's theorem [8] which establishes, under a certain set of (apparently) intuitive axioms, that any measure of belief can be mapped to a probability measure. The intuitiveness of the axioms and additional implicit assumptions made in the proof have been criticized in a number of ways and the proof has been modified to account for some of these [9]. Such proofs are most strongly favoured in the objective school, since they establish the universal applicability of probabil-

ity to belief. De Finetti, an actuary by training, argues along different lines, that a coherent and rational method of computing plausibility must be such that by offering odds on all possible outcomes according to the computed beliefs, a subject is not exposed to certain loss through some series of bets (a *Dutch book*) [6]. He established that the axioms of probability, and thus Bayesian inference, can be established as a system to do this (although not necessarily the only one [10] except under additional assumptions). Such arguments put prediction at the heart of probabilistic inference.

It is worth noting that Bayesian inference is not able to reach meaningful conclusions about problems involving *all* types of uncertainty. Bayesian inference rests on probability theory which in turn rests on the idea that it is possible to assign numerical values to the probability of the occurrence of events. However, this does *not* cover all types of uncertainty and in particular does not cover uncertainty when there is no underlying fact. For example, the statement "John behaved maturely" is a subjective statement that does not admit probabilistic interpretation [11]. It may be argued that the *vagueness* of this statement can be removed by defining the meaning of 'mature', but it is obvious that, by requiring further clarification, there is a type of uncertainty about which probability cannot reason, in this case the sense of the word 'mature'. Non-probabilistic uncertainty is not considered in this thesis, however approaches do exist for reasoning in such situations; see e.g. [11] and the references therein.

Nevertheless, Bayesian inference is a practically successful, coherent and often tractable way of dealing with the types of uncertainty that *can* be quantified as probability. The central component of any scheme for reasoning about this sort of uncertainty is a means of updating beliefs in the light of evidence. In Bayesian inference this update rule is given by Thomas Bayes' eponymous 1763 theorem [12] and its subsequent generalization by Pierre-Simon Laplace [13], which can be summarized in modern notation

as

$$p(X \mid Y) = \frac{p(Y \mid X)p(X)}{p(Y)}, \tag{1.1}$$

where $p(X)$ is the probability of an event X and $p(X \mid Y)$ is the conditional probability of an event X given that an event Y has occurred. This is a simple consequence of the definitions of probability and can be seen quite easily with the aid of a Venn diagram. If X is interpreted as a set of properties of interest, and Y is interpreted as a set of observations related to those properties then this theorem allows *prior* beliefs $p(X)$ about those properties to be updated using evidence arising from observations to give new *posterior* beliefs $p(X \mid Y)$ about the state of X after having made observations Y.

As well as the prior belief $p(X)$, calculation of the probability distribution $p(X \mid Y)$ using equation (1.1) requires a way of determining the *likelihood* of the observations given a certain set of properties $p(Y \mid X)$. Most Bayesian time series methods rely on a *model* of system behaviour in order to calculate this likelihood. A common approach is to assume that observed data is generated by a noisy process, for example the imperfect readings of a sensor, from an underlying true system state, which evolves over time according to a dynamical model. The true state of the system at any time is hidden, but observations reveal information that can be used to learn about that hidden state. In many cases, it is not necessary to calculate the observation probability $p(Y)$ because, since probabilities must integrate to 1, it can be treated as a normalizing constant of the probability distribution $p(X \mid Y)$.

Often, the observation generating model used in likelihood calculation is not truly known. In such cases the model chosen can itself be thought of as part of the prior assumptions of the inference system. The fidelity of the observation generating model to the true system can have a substantial impact on the accuracy of inference that can be made. If swans are assumed always white, and crows always black, encountering a black swan might

well lead to incorrect conclusions (i.e. that what was seen was, in fact, a crow). It is therefore important to model the behaviour of the underlying system as closely as possible. However, if this behaviour, or aspects of it, are unknown, either because parameters of the model are unknown or because the model itself is obscure it is desirable to incorporate this uncertainty into the inference process. In some cases, such as when trying to understand relationships between objects or the structure of an environment, aspects of the model structure themselves can be the properties of interest. Assuming that underlying systems exist and can be considered knowable in some sense, this uncertainty is a valid subject of Bayesian inference. By incorporating hidden system structure into the set of unknown system properties, the machinery of Bayesian inference can be used to learn these from the data. Unfortunately naive approaches to this very often lead to extremely difficult or intractable inference problems.

The aim of this work is to develop tractable methods for the inference of both hidden states and aspects of hidden model structure in some types of time series, and to do so within a Bayesian probabilistic framework. It is hoped that this will improve the accuracy of state inference, and will allow useful insights into the nature of the systems in question to be derived from observations. Motivating examples are taken from a number of application areas including finance, physical object tracking and audio restoration.

1.1 Thesis Outline

Chapter 2 briefly outlines some of the important techniques in the application of Bayesian inference to time series problems, with a particular focus on methods for the non-linear problems that occur most frequently throughout this work.

Chapter 3 examines the problem of state inference in jump-diffusion systems with linear diffusion dynamics through the use of the recently introduced variable rate particle filter. A general model for such systems is

developed and solved, and an efficient computational method for state inference is presented. As an example of the application of such methods, they are applied to the development of a trend-following system for finance.

Chapter 4 extends the work of chapter 3 to include estimation of model parameters for the jump-diffusion models studied there. Particle MCMC methods are developed for this problem, showing how the transdimensional filtering problem of estimating jump parameters can be cast in such a way as to be tackled with Particle MCMC methods, and these are compared to more standard reversible jump MCMC methods.

Subsequent chapters turn attention to the development of methods for systems with non-linear Langevin dynamics. As examples, the methods developed are applied to physical tracking applications. In the systems examined, the aim is to not only learn the hidden states of the model, but also to learn something about the structure of the system itself. Unlike standard parameter learning, fixed functional forms are not assumed for the structures being learnt. Instead, limited non-parametric assumptions are made and the form of these aspects of the systems in question are inferred from the data.

Chapter 5 introduces a non-linear model for the tracking of objects moving in a structured environment. The aim of the work is to simultaneously track objects moving throughout the environment and to learn the structure of the environment in which they are moving. This is closely related to simultaneous localization and mapping (SLAM) problems in robotics. In this case, the environment is modelled as an initially unknown potential field through which objects moves. By making only non-parametric Gaussian process prior assumptions about this field, a wide range of field shapes can be learnt from observations.

Chapter 6 extends the idea of object tracking to the multivariate case. Again the aim is to not just track the objects in question, but to learn something about their environment, in this case, the way they interact with each

other. A non-parametric method again based on Gaussian process priors is presented for this. In order to allow efficient solution it uses a bin-based approximation of the observations. The method is able to learn non-linear relationships and causality between objects, and a way of estimating the sparse structure of the network of relationships is given. It is related to methods used for inference in gene regulatory networks and is applicable in a broad range of settings.

Chapter 7 takes some of the sparse structure ideas from the preceding chapter and applies them to the removal of impulse noise in audio signals. A more efficient solution of a previously presented method is given.

Chapter 8 draws some overall conclusions and makes recommendations for future work.

1.2 Main Contributions

This thesis presents a number of results that, it is believed, extend the current state of the art in several areas. Along with a number of less significant innovations, the main contributions of this thesis are as follows.

- An alternative method to reversible jump MCMC for parameter inference in linear jump-diffusion systems, based on Particle MCMC methods (chapter 4)

- A new approach to non-parametric mapping in a SLAM-style problem, with a particular emphasis on tracking objects in structured environments. The approach uses a non-parametric Gaussian process prior model to learn a potential field model of the environment in which a target moves. Efficient solution methods based on Rao-Blackwellized particle filters are given for this model (chapter 5)

- A new way of learning non-linear inter-object relationships in multi-object systems is developed, allowing complicated inter-object dynamics to be learnt and causality between objects to be inferred. Again

based on non-parametric Gaussian process prior assumptions, the method allows the identification of a wide range of relationships between objects with minimal assumptions and admits efficient solution, albeit in batch form at present (chapter 6)

Some further contributions include

- An efficient Rao-Blackwellized particle filter for state inference in jump-diffusion models (chapter 3)
- A bin-based approach to sparse Gaussian process learning that is particularly suited for use in certain types of model and with certain inference methods (chapter 6)
- An algorithm for the removal of impulse noise (pops and clicks) in audio recordings (chapter 7)

Chapter 2

Bayesian Inference For State Space Models

This chapter gives a brief overview of some of the important techniques for Bayesian inference for state space models, with an emphasis on non-linear Markovian time-series models. Section 2.1 introduces the main concepts of state space models and reviews in outline some of the inference techniques that can be applied, such as sequential and batch state estimation (filtering and smoothing, respectively) and parameter estimation. It also gives details of two special cases in which state estimation is tractable: the finite state space case, and the linear Gaussian case. Section 2.2 introduces Markov chain Monte Carlo (MCMC) methods which are a fundamental tool in Bayesian state space analysis, especially in the batch case. Such methods are especially useful for parameter estimation and also form useful components in sequential Monte Carlo methods. Section 2.3 examines sequential Monte Carlo methods, also known as particle filters in the time series context. These provide a powerful set of methods for online state estimation in non-linear state space models.

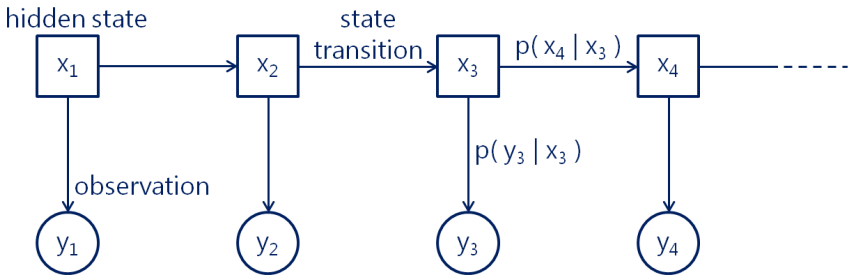


Figure 2.1: The structure of Markovian state space models for time series. The x variables represent the latent state of the system at each observation time and the y variables represent observations made of the system. Arrows represent causal relationships between variables; it is also common to show this system as the undirected graph given by replacing the arrows with undirected edges. In this case that is equivalent (describes the same conditional independence relationships), as is the directed graph with the edges between x_t and x_{t+1} reversed

2.1 State Space Models

State space models are common models of time series, based on the assumption of an underlying system that evolves in a definable way, from which observations are generated through another definable process. Both the system evolution and observation generation are most usually defined probabilistically, leading to a probability distribution over subsequent states and observations given the system's history to that point. A common assumption is that the system evolution can be described as a *Markov* process, in which subsequent states are conditionally independent of all preceding states given the current one. Their simpler mathematical description makes inference in Markov systems much more tractable than in the general case, but a large class of useful models can still be described by such systems. In sequential applications Markov models also benefit from reduced storage overhead, because it is unnecessary to store any but the current state.

Figure 2.1 shows the structure of the type of Markov models used for time series modelling. Observations at any given time are assumed to depend only on the current system state. The transition between states x at

one observation time to the next is described by a *state transition function* in each period, giving the *dynamical model* of the system. Similarly, the generation of observations y from the state is described by an *observation function*. These functions can change with time, and, in the models dealt with here are probabilistic, meaning that the transition function from one period to the next can be described as a conditional probability distribution $p(x_{t+1} \mid x_t)$ giving the probability density of the next state x_{t+1}, conditional on the current state x_t. Similarly the observation model can be expressed as a conditional probability density $p(y_t \mid x_t)$ of the observation y_t given the current system state x_t. Discrete or continuous valued observations and states (or a mixture of both) can be incorporated in these models, as can multi-dimensional states and observations, giving rise to multivariate transition and observation densities. For example, in a simple tracking model, the state might be composed of an object's position and velocity. This might be augmented with a discrete indicator variable indicating whether the object was applying its own internal thrust in each period.

Observations do not directly reveal the system state, which is hidden (or *latent*), hence such models are often known as *hidden Markov models* (HMMs). A very common aim in constructing these models is to learn about this hidden state from a sequence of observations. An obvious way to do so is via the probability of the hidden states conditioned on the received observations, $p(x_{1:t} \mid y_{1:t})$, where $x_{1:t}$ is used as shorthand for the set $\{x_1, x_2, ..., x_t\}$ and similarly $y_{1:t}$. For the Markov models described, this distribution can be calculated from the transition and observation functions as

$$p(x_{1:t} \mid y_{1:t}) \propto p(x_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1}) \prod_{t=1}^{T} p(y_t \mid x_t),$$

using Bayes' theorem and the the conditional dependence structure shown in figure 2.1. The distribution $p(x_1)$ is a *prior* belief on the distribution of

x_1, and the distribution $p(x_{1:t} \mid y_{1:t})$ is the *posterior* distribution of the latent states given the observations. The constant of proportionality is given by $p(y_{1:t})^{-1}$; this is relegated to the proportionality constant because it is the same for all possible state sequences $x_{1:t}$, and is a normalizing constant for the probability distribution over $x_{1:t}$. The problem of state estimation is then that of calculating this distribution, either exactly or approximately. In some cases, a point estimate of the state is required and, in that case, the state sequence that maximizes this distribution, the *maximum a posteriori* (MAP) state estimate, is a good candidate.

A special case of the state estimation problem is the *filtering* problem in which the aim is to determine the current state of the system, given observations up to that moment. In probabilistic terms, filtering is concerned with the distribution $p(x_t \mid y_{1:t})$. A further special case is *fixed-lag smoothing*, in which the system state is to be estimated after seeing some fixed amount L of further observations; the corresponding distribution of interest is $p(x_{T-L} \mid y_{1:T})$. The estimation of one or an entire sequence of states from a given set of observations extending up to or beyond the end of the state sequence being estimated is called *fixed-interval smoothing* and the corresponding distribution is $p(x_{t_1:t_2} \mid y_{1:T})$ with $1 \leq t_1 \leq t_2 \leq T$. A further related problem is that of *prediction*, in which the aim is to determine the state of the system at times beyond the last observation. The corresponding distribution is $p(x_t \mid y_{1:s})$ with $t > s$.

Inference for state space models can be tackled in a number of ways, as briefly outlined in the rest of this section. There are two important special cases for which exact inference methods exist. These are finite state space models, in which all state variables take one of a finite number of values, and linear Gaussian models, in which the state and observations are continuous-valued, and where transition and observation functions take the form of linear transformations of the current state, distorted by additive Gaussian noise. These are dealt with in the following two sections.

2.1.1 Finite State Spaces

If the state space of the model is finite, that is, the system state can only be in one of a finite number of values at each point in time, a number of efficient exact algorithms exist for state inference, including ones to find the filtering and (marginal) smoothing distributions and one to find the MAP state sequence.

Current State Distribution - Forward Filter

For finite state space models, the filtering distribution $p(x_t \mid y_{1:t})$ can be found exactly, albeit in quadratic time in the number of states in the state space. As with many filtering algorithms, the easiest formulation is as a *prediction-correction* algorithm, with a first step making a prediction $p(x_t \mid y_{1:t-1})$ of the next state given only the current observations, and the following step 'correcting' this prediction using the incoming observation to give $p(x_t \mid y_{1:t})$. The prediction step can be formulated as an integration over all current states, evaluating the probability of arriving at each time t state by any possible route via a time $t-1$ state, so that

$$p(x_t \mid y_{1:t-1}) = \int_{S_{t-1}} p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1}. \tag{2.1}$$

The distribution $p(x_{t-1} \mid y_{1:t-1})$ is the posterior filter distribution after the previous observation. In finite state spaces the predictive distribution is given by a finite sum over the state space at $t-1$

$$p(x_t \mid y_{1:t-1}) = \sum_{S_{t-1}} p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}),$$

where, because every possible state at $t-1$ must be considered as a route to each successor state at t, this operation is quadratic in the size of the state space.

The predictive probability for each state can be updated in light of the observation at time t using an application of Bayes' theorem and the con-

ditional probability structure from figure 2.1:

$$\begin{aligned} p(x_t \mid y_{1:t}) &= \frac{p(y_t \mid x_t)p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ &\propto p(y_t \mid x_t)p(x_t \mid y_{1:t-1}). \end{aligned} \tag{2.2}$$

The denominator in the first line does not depend on the state x_t and so can be treated as a normalizing constant. Since $p(x_t \mid y_{1:t})$ is a probability distribution, its sum over all possible states must be 1, and therefore the filter probabilities can be found by normalizing the values found using equation (2.2) so that they sum to 1.

Smoothing Distribution - Forward-Backward Algorithm

To calculate marginal smoothing distribution $p(x_t \mid y_{1:T})$, a backward pass can be added to the forward filter. The marginal smoothing distribution can be written as the product of the forward predictive density used during filtering and the likelihood of the future observations given x_t:

$$\begin{aligned} p(x_t \mid y_{1:T}) &= p(x_t \mid y_{1:t-1}, y_{t:T}) \\ &\propto p(x_t \mid y_{1:t-1})p(y_{t:T} \mid x_t), \end{aligned} \tag{2.3}$$

with $p(y_{t:T})^{-1}$ as the constant of proportionality. For $t < T$, this backward likelihood can be found via the recursion

$$p(y_{t:T} \mid x_t) = p(y_t \mid x_t) \int_{S_{t+1}} p(y_{t+1:T} \mid x_{t+1})p(x_{t+1} \mid x_t)dx_{t+1}.$$

In the finite state case this integral can again be calculated as a finite sum with quadratic complexity in the size of the state space (because all values of x_{t+1} must be considered for each value of x_t). The distribution $p(y_{t+1:T} \mid x_{t+1})$ is this backward distribution from the previous step, so the integral can be calculated recursively, with $p(y_T \mid x_T)$ given by the final filter distribution from the forward step. By combining the forward and backward distributions as shown in equation (2.3) and normalizing, the

marginal smoother distributions for each state can be found.

In the finite state space context, this algorithm is known as the forward-backward algorithm. It is an instance of a *two-filter smoothing formula* [14; 15], with a forward filter calculating $p(x_t \mid y_{1:t})$, and another separate filter sometimes known as a *backward information filter*, working backwards from T to calculate $p(y_{t+1:T} \mid x_t)$.

MAP State Sequence - Viterbi Algorithm

Because it calculates the marginal smoother distributions, the forward-backward algorithm cannot be used to give the most likely state *sequence*. However, this can be obtained, again in time quadratic in the size of the state space, using the Viterbi algorithm [16], which was first developed as a decoding algorithm for convolutional codes over noisy channels. It is a forward-recursive dynamic programming algorithm that, at each observation time t, determines the highest probability sequence of states ending in each state $s_t \in S_t$, i.e. finding

$$\tilde{x}_{1:t}^{s_t} = \text{argmax}_{x_{1:t-1}} p(x_{1:t-1}, x_t = s_t \mid y_{1:t}),$$

along with the probability of these paths, or something proportional to it $k_t p(\tilde{x}_{1:t}^{s_t} \mid y_{1:t})$, where k_t is constant. If a set of such paths and probabilities are known at time t, then the equivalent paths can be found at time $t+1$ by noting that each of these must follow one of the previous maximum probability paths up to time t, since there is no higher probability route to the ancestor of the current state, whatever that might be. The maximum probability path to each state can be found by evaluating the probability of getting to the state via each of the maximum probability paths from the previous step, i.e. by evaluating

$$\begin{aligned} p(\tilde{x}_{1:t}^{s_t}, x_{t+1} = s_{t+1} \mid y_{1:t+1}) \quad &\propto \quad p(\tilde{x}_{1:t}^{s_t} \mid y_{1:t}) p(y_{t+1} \mid x_{t+1} = s_{t+1}) \\ &\quad \times p(x_{t+1} = s_{t+1} \mid x_t = s_t), \end{aligned}$$

for each s_t and s_{t+1} pair. For each s_{t+1} the route with maximum probability is chosen and stored along with its probability (to proportionality). At any time of interest t, the maximum probability path is simply the path to the value s_t with the highest probability.

Approximations

Quadratic complexity in the number of states might lead to problems that are practically intractable in reasonable time for problems with large state spaces. In this case approximate algorithms will still be required. For example, a finite state version of the particle filter can be used in finite state spaces for approximate state inference.

2.1.2 Linear Gaussian Models

Another important special case (used extensively in this thesis) is the continuous-valued case with transition and observation functions that can be described as linear transforms of the current state subject to additive Gaussian noise and a known control signal. These systems have the form

$$\begin{aligned} p(x_t \mid x_{t-1}) &= F_t x_{t-1} + B_t u_t + \epsilon_t \\ p(y_t \mid x_t) &= H_t x_t + \eta_t \end{aligned}$$

where $\epsilon_t \sim \mathcal{N}(0, Q_t)$ and $\eta_t \sim \mathcal{N}(0, R_t)$ and where H_t, F_t and B_t are known matrices and u_t is a known signal.

Kalman Filter

In the linear Gaussian case marginal filtering distributions can be calculated exactly as a series of Gaussians marginal posterior distributions. This is known as the Kalman filter [17]. It can be derived straightforwardly as a Bayesian prediction-correction method in the same way as the filter in the finite state case [18]. In this case, the integral in equation (2.1) can be solved in closed form as long as the previous filter distribution is also Gaussian by

using identity (A.2) in appendix A. Assuming a previous filter density of $p(x_{t-1} \mid y_{1:t-1}) \sim \mathcal{N}\left(\mu_{t-1|t-1}, \Sigma_{t-1|t-1}\right)$, this gives the Gaussian predictive density for x_t

$$p(x_t \mid y_{1:t-1}) \sim \mathcal{N}\left(\mu_{t|t-1}, \Sigma_{t|t-1}\right)$$

with

$$\mu_{t|t-1} = F_t \mu_{t-1|t-1} + B_t u_t \tag{2.4}$$

$$\Sigma_{t|t-1} = F_t \Sigma_{t-1|t-1} F_t' + Q_t. \tag{2.5}$$

Using equation (2.2), a further application of identity (A.2) and some rearrangement using the Woodbury matrix identity, these predictions can be corrected using the time t observation to give the posterior state distribution at t as

$$p(x_t \mid y_{1:t}) \sim \mathcal{N}\left(\mu_{t|t}, \Sigma_{t|t}\right)$$

where

$$\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - H_t \mu_{t|t-1}) \tag{2.6}$$

$$\Sigma_{t|t} = (I - K_t H_t)\, \Sigma_{t|t-1} \tag{2.7}$$

$$K_t = \Sigma_{t|t-1} H_t' \left(H_t \Sigma_{t|t-1} H_t' + R_t\right)^{-1}. \tag{2.8}$$

That this posterior is itself Gaussian is an example of the self-conjugacy of the Gaussian distribution. The requirement that the preceding posterior state distribution $p(x_{t-1} \mid y_{1:t-1})$ be Gaussian therefore holds as long as the initial prior distribution $p(x_1)$ is itself Gaussian. In fact, the Kalman filter does not *rely* on Gaussian noise and, even without this assumption, can be shown to provide the optimal *linear* estimator of the state in the squared error sense [17].

A useful additional piece of information that can be derived from the

Kalman filter is the likelihood of the observations $p(y_{1:T})$ via the *prediction error decomposition* (PED) [19]. The likelihood can be decomposed as

$$p(y_{1:T}) = p(y_1) \prod_{t=1}^{T-1} p(y_t \mid y_{1:t-1}),$$

and the individual terms in the product calculated as

$$p(y_t \mid y_{1:t-1}) = \int p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) dx_t.$$

Since the predictive state distributions $p(x_t \mid y_{1:t-1})$ and the observation densities $p(y_t \mid x_t)$ are both Gaussian, this integral can be calculated in a similar way to that in equation (2.1) for the predictions, giving

$$p(y_t \mid y_{1:t-1}) \sim \mathcal{N}(\mu_{y_t}, \Sigma_{y_t}) \tag{2.9}$$

where

$$\begin{aligned} \mu_{y_t} &= H_t \mu_{t|t-1} \\ \Sigma_{y_t} &= H_t \Sigma_{t|t-1} H_t' + R_t. \end{aligned}$$

This calculation is particularly useful in parameter estimation, when it can be used to evaluate the likelihood of the observations under different system parameters $p(y_{1:T} \mid \theta)$ for various θ.

The Kalman filter also offers the simplest way of performing fixed-lag smoothing in linear Gaussian systems, which can be achieved by augmenting the state space with the state at previous times so that $x_t^+ = [x_t \; x_{t-1} \; \ldots \; x_{t-s}]'$. The state transition and observation matrices are then augmented to be of the form

$$F_t^+ = \begin{bmatrix} F_t & 0_{d \times s} \\ I_{ds} & 0_{ds \times 1} \end{bmatrix}, \qquad H_t^+ = \begin{bmatrix} H_t & 0_{p \times ds} \end{bmatrix},$$

where I_n is an $n \times n$ identity matrix, $0_{m \times n}$ is an $m \times n$ matrix of zeros, d

is the dimensionality of x_t and p is the dimensionality of the observations. This gives a system in which the desired fixed-lag smoothing estimate is given by the appropriate element of the augmented filter solution.

Fixed Interval Smoother

Fixed interval smoothers in the linear Gaussian case can be derived in several ways, including via the two-filter formulation in equation (2.3). An alternative approach that leads to some of the most popular smoothers is via the *forward filtering, backward smoothing* recursion [20; 21; 15]. This relies on writing the smoothing distribution in terms of the filtering distribution at t and the smoothing distribution at $t+1$ (which is assumed available), giving

$$\begin{aligned} p(x_t \mid y_{1:T}) &= \int p(x_t \mid x_{t+1}, y_{1:t}) p(x_{t+1} \mid y_{1:T}) dx_{t+1} \\ &= p(x_t \mid y_{1:t}) \int \frac{p(x_{t+1} \mid x_t)}{p(x_{t+1} \mid y_{1:t})} p(x_{t+1} \mid y_{1:T}) dx_{t+1}. \quad (2.10) \end{aligned}$$

The first line makes use of the conditional independence relationship shown in figure 2.1 to write $p(x_t \mid x_{t+1}, y_{1:T}) = p(x_t \mid x_{t+1}, y_{1:t})$. In the linear Gaussian case, this is tractable, since it can be written

$$p(x_t \mid y_{1:T}) = \mathcal{N}\left(x_t; \mu_{t|t}, \Sigma_{t|t}\right) \int \frac{\mathcal{N}\left(x_{t+1}; F_t x_t, Q_t\right)}{\mathcal{N}\left(x_{t+1}; \mu_{t+1|t}, \Sigma_{t+1|t}\right)} \mathcal{N}\left(x_{t+1}; \mu_{t+1|T}, \Sigma_{t+1|T}\right) dx_{t+1},$$

where $\mu_{t+1|T}$ and $\Sigma_{t+1|T}$ are the mean and variance of the smoothing distribution of x_{t+1} given observations to T (simply the final filter distributions when $t = T-1$). Using identities (A.2) and (A.3) from appendix A, several applications of the Woodbury matrix identity and noting that a Gaussian integrated over its arguments evaluates to 1 (along with some tedious algebra), it is possible to arrive at the Rauch-Tung-Striebel fixed interval smoother [20], given by

$$p(x_t \mid y_{1:T}) = \mathcal{N}\left(\mu_{t|T}, \Sigma_{t|T}\right)$$

with, for $t < T$,

$$\begin{aligned}
\mu_{t|T} &= \mu_{t|t} + C_t \left(\mu_{t+1|T} - \mu_{t+1|t}\right) \\
\Sigma_{t|T} &= \Sigma_{t|t} + C_t \left(\Sigma_{t+1|T} - \Sigma_{t+1|t}\right) C_t' \\
C_t &= \Sigma_{t|t} F_t' \Sigma_{t+1|t}^{-1}.
\end{aligned}$$

The algorithm starts at $t = T$ with the final filter distribution, and proceeds backwards to $t = 1$. Several variants of the smoother exist, including the Modified Bryson-Frazier smoother [22], which avoids inversion of the covariance matrix and so might in some cases be preferable. It is also possible to derive smoothers via the two-filter formulation seen in the Forward-Backward algorithm above, although care must be taken in this case to avoid unintegrable backward likelihoods; see [21; 15] for further details.

2.1.3 Filtering in the General Case

Other than in these special cases, the calculation of filtering and smoothing distributions is intractable and approximations are necessary. Essentially, all methods must approximate the integral in equation (2.1) and the multiplication of densities in equation (2.2), which amounts to a (possibly non-linear) transform of the state estimate, which is a random variable. The following sections attempt to briefly outline the most relevant of these approximations to this work.

For the filtering problem, an approach to cope with non-linear or non-Guassian systems is to modify the Kalman filter. The first such method, the *extended Kalman filter* (EKF) was introduced by NASA in order to model non-linearities in spacecraft dynamics at around the same time as the development of the Kalman filter [23]. The EKF estimates the first two moments of the state distribution and works by linearizing non-linear transition and observation functions about the current mean state estimate using their Taylor series expansions. In the EKF approximation, the mean of a nonlinear function of a random variable $h(x)$ is approximated as (in the

univariate case)

$$\begin{aligned} \mathbb{E}(h(x)) &= \int h(x)p(x)dx \\ &\approx \int \left(h(\bar{x}) + (x - \bar{x}) \frac{\partial h}{\partial x}\bigg|_{\bar{x}} \right) p(x)dx \\ &= h(\bar{x}) \qquad \text{if } \bar{x} = \mathbb{E}(x), \end{aligned} \tag{2.11}$$

where the contents of the brackets on the second line can be recognized as the Taylor series expansion of $h(x)$ about $\bar{x}$ truncated after two terms. The variance is given similarly (when $\bar{x} = \mathbb{E}(x)$) as

$$\mathbb{V}(h(x)) = \left(\frac{\partial h}{\partial x}\bigg|_{\bar{x}} \right)^2 \mathbb{V}(x). \tag{2.12}$$

With additive Gaussian noise, the EKF turns out to be identical to the Kalman filter described in section 2.1.2, but with a modified prediction mean

$$\mu_{t|t-1} = f(\mu_{t-1|t-1}, u_{t-1})$$

where f is the non-linear state transition function, and the F_t and H_t matrices in the other calculations replaced with the Jacobian matrices of the respective functions at the current state estimate, so that

$$F_t = \frac{\partial f}{\partial x}\bigg|_{\mu_{t|t}, u_t} \qquad H_t = \frac{\partial h}{\partial x}\bigg|_{\mu_{t|t-1}}$$

where h is the observation function.

The EKF is simple to implement and, if calculation of the Jacobians is not too complicated, takes similar computational effort to the standard Kalman filter. However, the method has several drawbacks. In some cases, calculation of the Jacobians can be difficult and they are not even guaranteed to exist, for example if the functions f and h contain discontinuities. Even worse, in highly non-linear cases where the system is not well approximated by the first two terms in the Taylor series, serious er-

rors can arise, with the EKF estimates diverging from the true values. [24] gives an analysis of this, and notes that one such case where linearization provides a biased, inconsistent estimator is that of transforming from polar to Cartesian coordinates, a very important transform in tracking problems due to the range and bearing output of many sensors. Further terms can be considered in the Taylor series (for example, [25] considered the first two derivatives), although this rapidly increases the complexity of the method.

A fairly recent alternative to the EKF is the *unscented Kalman filter* (UKF) first proposed in [26] and treated in more detail in [27] and [24]. This overcomes a number of the problems of the EKF by introducing an *unscented transform* in order to approximate the required non-linear densities and is based on the idea "that it is easier to approximate a probability distribution than it is to approximate an arbitrary non-linear function" [24]. Like the EKF it approximates the required distributions using their first two moments. It uses a weighted collection of *sigma points* x^i chosen in sample space so that their mean and covariance is that of the current state estimate and then applies the required non-linear function to each of these points. This yields a transformed set of points, the sample mean and covariance of which can be taken as the required estimates. This is equivalent to approximating the integral in equation (2.11) as

$$\mathbb{E}\left(h(x)\right) \approx \sum_i w_i h(x^i), \tag{2.13}$$

with the x^i chosen such that $p(x) \approx \sum_i w_i \delta_{\{x^i\}}$, with this approximation in particular matching the first two moments of $p(x)$. The variance is approximated similarly as

$$\mathbb{V}\left(h(x)\right) = \sum_i w_i \left(h(x^i) - \bar{h}\right)^2,$$

where $\bar{h}$ denotes the mean approximation given by equation (2.13). As will be seen below, this is similar to the approximation used by the particle filter, but there are important differences. Most importantly, the sigma point ap-

proximation is not used to represent the target densities in the UKF, rather it is used to estimate the first two moments of the target densities; it is these moments that are used to represent the densities. Another difference is that the choice of sigma points is deterministic, and their weights can be negative, making the UKF more akin to traditional quadrature methods for the approximation of integrals. Indeed, the integral approximation used by the UKF is closely linked to Gauss-Hermite quadrature rules [28; 29]. The choice of sigma points is flexible, but the general scheme suggested by [26] is to have $2d+1$ sigma points in d-dimensional space, located up and down each of the principle axes of the covariance matrix of x from its mean, with an additional point at the mean itself. [30] and [24] gives further guidance on choosing sets of sigma points to mitigate higher-order effects.

The UKF has been found to outperform the EKF in terms of error for a large number of problems e.g. in [27; 31; 24] involving significant non-linearities in the state transition or observation functions. The method can be shown to be accurate for Gaussian inputs to the third order Taylor series terms for all non-linearities [24] and to at least the second order for non-Gaussian inputs [30; 27]. This, combined with the relatively small number of sigma points required makes the method a good choice in a lot of computationally limited situations.

Another important approach that has received much attention in recent years is the particle filter. Like the UKF, this also relies on a sample-based approximation of distributions and integrals, but in this case the approximation uses importance sampling to draw the samples randomly from an importance distribution q so that $x^i \sim q(x)$ (with q having the same support as p or, more technically, that p and q define equivalent probability measures). The expectation of a function of x is thus given by the approx-

imation

$$\mathbb{E}(h(x)) = \int h(x)\frac{p(x)}{q(x)}q(x)dx \tag{2.14}$$

$$\approx \sum_i h(x^i)\frac{p(x^i)}{q(x^i)}. \tag{2.15}$$

This approximation has the nice property that as the number of samples approaches infinity, the approximation approaches the true expectation asymptotically. The variance is defined similarly. In this context the samples x^i are called *particles* and the product $w_i = \frac{p(x^i)}{q(x^i)}$ gives a 'weight' to each particle. It is clear that here, unlike in the UKF, weights cannot be negative.

The particle filter itself is built around sequential updates of these importance sampling approximations, which is achieved through increasing the dimensionality of the problem and sample space at each new time period. This makes the use of non-Markovian models easier with this method. Further details of the particle filter are given in section 2.3.

Because of its random sampling approach, the particle filter generally has a high computational cost, because many particles are required in order to give good representations of the distributions they approximate, requiring many evaluations of the importance, transition and observation functions. On the other hand, the particle filter allows the approximation of arbitrary densities to arbitrary precision thanks to its asymptotic convergence to the true distributions of interest. Thus, the choice of filtering method is, as ever, a compromise that must be assessed for the application at hand.

2.1.4 Smoothing in the General Case

There are several approaches to fixed interval smoothing in the general case, both approximate and asymptotically exact (in that they will asymptotically approach the true smoothing distribution as the effort expended on them approaches infinity). Smoothing algorithms can be built around the approximate filtering techniques from the section above using two ap-

proaches: the two-filter approach seen earlier in the Forward-Backward algorithm in section 2.1.1, and the forward-filtering backward-smoothing approach seen in deriving the RTS smoother in section 2.1.2.

Smoothing methods based on the EKF generally use a modified version of the backward-smoothing steps found in the RTS smoother, identical to those in section 2.1.2, except with F_t' replaced with the transpose of the Jacobian of the state transition function at $\mu_{t|t}$. This gives a forward-filter backward-smoother approximation based around linearization of the transition and observation functions. Recently, [32] introduced a forward-filtering backward-smoothing algorithm based around the UKF, using a similar formulation to that of the RTS smoother, and which is shown to perform equally well to the two-filter version. It relies on a Gaussian assumption about the shape of the approximating distributions; this is *not* the case for the UKF [24], but is a common assumption when using the UKF in practice [27].

Two-filter versions of the UKF are given in [29] and [21]. Such two-filter smoothing methods are discussed in detail in [21; 15]. There it is noted that care must be taken in developing these methods due to the target distribution of the backward smoother $p(y_{t:T} \mid x_t)$ *not* being a probability distribution for x_t and, in some cases the integral of this with respect to x_t may be unbounded. This can make the formulation of methods that propagate this non-probability (and possibly non-finite) measure correctly tricky in the general case. In [21; 15] this is tackled by the introduction of an artificial prior $\gamma(x_0)$ on x_0, along with an artificial recursive formula to allow the calculation of priors $\gamma(x_t)$ on subsequent x_t. This allows the artificial probability distribution $\tilde{p}(x_t \mid y_{t:T}) \propto p(y_{t:T} \mid x_t)\gamma(x_t)$ to be used and propagated in place of $p(y_{t:T} \mid x_t)$ and thus the standard EKF and UKF methods to be applied directly to the backward filtering problem. See section 2.3.3 for additional details.

Similarly, it is also possible to create fixed-interval smoothers based around the particle filter via both the forward-filtering backward-smoothing

and two-filter formulations. These are covered in more detail in section 2.3.3. Some versions of these smoothers have the advantage that the samples drawn are from the joint posterior over the state space $p(x_{1:T} \mid y_{1:T})$ rather than just from the marginal distributions $p(x_t \mid y_{1:T})$.

A different approach to smoothing is to use a Markov chain Monte Carlo (MCMC) algorithm with the required smoothing distribution as its target. Section 2.2 covers MCMC methods in more detail, but it straightforward to set up such an algorithm in most cases. This is most usually done via Gibbs sampling allowing the state at time t to be sampled from its conditional distribution

$$p(x_t \mid x_{1:t-1}, x_{t+1:T}, y_{1:T}) \propto p(x_t \mid x_{t-1})p(x_{t+1} \mid x_t)p(y_t \mid x_t). \tag{2.16}$$

It is also possible to sample from the conditional joint distribution of the state over several periods; see sections 6.6.1 and 6.6.2 for examples. Though easy to set up, such methods suffer from a couple of important problems. Neighbouring states can be closely correlated, especially when the state transition model does not contain much noise. This limits the size of individual moves meaning that mixing (the speed with which the sampler explores the posterior) can be slow. Due to this, it can also be difficult for these methods to escape local maxima in the posterior. Some strategies to avoid such problems are covered in section 2.2.3.

A recent alternative to MCMC methods for smoothing, covered in more detail in chapter 4 is the Particle Independence Metropolis Hastings (PIMH) sampler introduced by [33]. This is a hybrid method that uses a particle filter to draw samples $x'_{1:T}$ of the state and evaluate their approximate likelihood $Z' \approx p(y_{1:T} \mid x'_{1:T})$. It can then be shown that, by accepting this proposal with probability $\min(1, Z'/Z)$, where Z is the approximate likelihood of the current sample, exact samples can be drawn from the smoother distribution; see section 4.3.2. Like some versions of the particle filter based smoother, this method is also able to draw samples from the joint state distribution.

In some cases state spaces have variable dimension, for example if the state is composed of an unknown number of events occurring at random times. Both MCMC and PIMH can deal with this situation. In the case of MCMC, this is via reversible-jump MCMC introduced in [34] (see section 2.2.3), whereas in the PIMH case, it can be done through the use of variable rate particle filters [35] as shown in chapter 4 for an example with a partially variable-dimensioned state space.

2.1.5 Parameter Estimation

The definitions of transition and observation functions along with their noise properties in state space models, typically involve a number of parameters. In some cases these are known in advance, but in many cases they are not and it is desirable to estimate them from the data. In some cases, estimation of these parameters is the primary objective of the model. In a Bayesian setting, the parameter estimation problem corresponds to determining the distribution

$$p(\theta \mid y_{1:T}) \propto p(y_{1:T} \mid \theta)p(\theta),$$

where $p(y_{1:T} \mid \theta)$ is the *likelihood* of the observations given the parameter and $p(\theta)$ is a prior distribution on the parameter value. In a non-Bayesian setting, maximum likelihood methods that attempt to find the value of θ maximizing $p(y_{1:T} \mid \theta)$ are commonly used to find point estimates of θ. The Bayesian 'equivalent' are MAP methods that attempt to find values of θ maximizing $p(\theta \mid y_{1:T})$. In fact, there is a fundamental philosophical difference between Bayesian and Frequentist approaches to parameter estimation, see e.g. chapter 37 of [36]; the latter are not considered further here.

Batch Parameter Estimation

Numerous methods for both approximate and (asymptotically) exact batch parameter estimation have been developed. They can be broadly divided

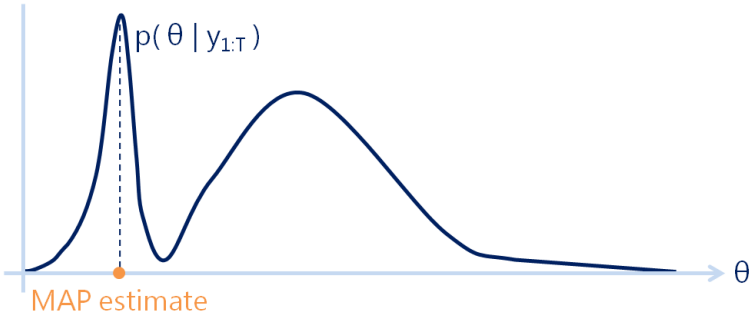


Figure 2.2: The risk of MAP estimation is that it can find an estimate with high posterior density that does not correspond well to the *region* with the greatest posterior probability

into point estimation techniques that aim to find MAP estimates of θ and fully Bayesian approaches that aim to determine the distribution $p(\theta \mid y_{1:T})$. Point estimates are often much easier to calculate and to use in subsequent calculations. However, there is a danger such point estimates can find solutions that give high posterior density values, but which do not lie in the areas containing most of the probability mass, as illustrated in figure 2.2.

In some simple cases the posterior is analytically tractable and so MAP parameter estimates can be found by analytical optimization of the posterior with respect to θ. Alternatively, if the posterior can be evaluated at a point θ, numerical optimization methods can be used to find MAP estimates. Such methods are generally applicable, but can be computationally expensive if the likelihood is difficult to evaluate. There are many possible optimization approaches, including methods such as gradient ascent and conjugate gradient methods for unimodal posteriors. For multimodal posteriors, such methods risk becoming stuck at a sub-optimal local maximum, so methods such as simulated annealing [37] or genetic algorithms may be required to attempt to find global optima.

A useful method for MAP estimation that is very often applicable for state space models is the popular *Expectation Maximization* (EM) algorithm, originally proposed in [38]. This allows the problem of parameter estimation to be tackled iteratively via a two step process by introducing a set of

hidden states x, the distribution of which should be easy to find given the observations and parameters. In state space models these naturally correspond to the hidden model states $x_{1:T}$ and the algorithm then consists of a so-called E-step, which finds the distribution $p(x_{1:T} \mid y_{1:T}, \theta^{(i)})$ where $\theta^{(i)}$ is the current estimate of θ, and an M-step that generates a new estimate of θ by maximizing a function of θ that turns out to be $\mathbb{E}_{x_{1:T}}(\log p(x_{1:T}, y_{1:T} \mid \theta)) + \log(p(\theta))$ over the $x_{1:T}$ distribution found in the E-step. The idea is that each of these steps should be much simpler than direct optimization of $p(\theta \mid y_{1:T})$ and by repeated iteration a (local) maximum will be reached. Following [39], this can be seen by writing the log-posterior in terms of a distribution $q(x)$ over the hidden states, and the parameter θ:

$$\log p(\theta \mid y) \quad = \quad \mathcal{L}(q, \theta) + \mathrm{KL}(q \| p(x \mid y, \theta)), \tag{2.17}$$

where $\mathrm{KL}(q \| p(x \mid y, \theta))$ is the Kullback-Liebler divergence between distributions q and $p(x \mid y, \theta)$, defined as

$$\mathrm{KL}(q \| p(x \mid y, \theta)) = \int q(x) \log \frac{q(x)}{p(x \mid y, \theta)} dx \geq 0.$$

This is a non-symmetric measure of the difference between two probability distributions [40]. It cannot be negative and is zero when the two distributions are identical. Thus, $\mathcal{L}(q, \theta)$ must be a lower bound on $\log p(\theta \mid y)$, given by

$$\mathcal{L}(q, \theta) = \int q(x) \log \frac{p(x, y \mid \theta)}{q(x)} dx + \log p(\theta) + \log p(y).$$

This form for $\mathcal{L}$ can be verified by applying the the chain rule of probability $\log p(x, y \mid \theta) = \log p(x \mid y, \theta) + \log p(y \mid \theta)$ to the above expression. Since $\log p(\theta \mid y)$ is independent of q, equation (2.17) indicates that the bound $\mathcal{L}(q, \theta)$ can be maximized with respect to q for any given value of θ when the Kullback-Liebler divergence is minimized, i.e. when $q(x) = p(x \mid y, \theta)$. In state space models, this is given by the fixed interval smoothing

distribution and thus the smoothing algorithms of the preceding sections can be used to find it (perhaps approximately). This is the E-step of the algorithm.

Assuming that the distribution of x is found using the current parameter estimate $\theta^{(i)}$, the corresponding lower bound is given by

$$\begin{aligned}
\mathcal{L}(q,\theta) &= \int p(x \mid y,\theta^{(i)}) \log \frac{p(x,y \mid \theta)}{p(x \mid y,\theta^{(i)})} dx + \log \frac{p(\theta)}{p(y)} \\
&= \int p(x \mid y,\theta^{(i)}) \log p(x,y \mid \theta) dx - \int p(x \mid y,\theta^{(i)}) \log p(x \mid y,\theta^{(i)}) dx + \log \frac{p(\theta)}{p(y)} \\
&= \mathbb{E}_x \left(\log p(x,y \mid \theta) \mid y,\theta^{(i)} \right) + \log p(\theta) + \text{constant w.r.t. } \theta \qquad (2.18)
\end{aligned}$$

This expectation (plus the prior term) can be maximized with respect to θ to give a new estimate of θ that again increases the lower bound on $\log p(y \mid \theta)$. Thus, the E-step forms this expectation and the M-step maximizes it. By repeating these steps the algorithm will converge to a local maximum of $p(y \mid \theta)$ [41]. In fact, it is not necessary to maximize the expectation with respect to θ, only to increase the lower bound. This is helpful in situations where the maximization step is intractable, since it allows approximate or partial maximization to be used. Similarly, the expectation step also need only increase the lower bound. Algorithms making use of these partial optimization steps are called *generalized expectation-maximization* (GEM) algorithms. In non-linear cases, it might be impossible to guarantee to increase the lower bound when forming the expectation, due to the use of an approximate smoothing method. In this case *stochastic expectation maximization* (SEM) can be used; see section 2.3.3.

The algorithm outlined here does not specify the details of its application in practice, and the maximization step might be difficult; indeed, finding an analytic expression for the expectation might be difficult, depending on the complexity of the joint model and the form of $q(x)$. However, it gives an idea of how and when EM can be applied and EM algorithms exist for many useful cases. An EM algorithm for estimating the parameters of a linear Gaussian system (the state and observation matrices and the noise

covariance matrices) is given by [42]. For finite state space models, a GEM approach to parameter estimation is given by the Baum-Welch algorithm [43], which pre-dates the more general version of EM in [38]. [44] derives a GEM algorithm for non-linear system estimation based on the EKF and [45] develops one for jump Markov linear systems.

EM methods produce a single point MAP estimate for the parameters, but a fully Bayesian treatment requires estimation of the distribution $p(\theta \mid y_{1:T})$. A natural question is whether the EM framework can be extended to such estimation by forming a lower bound based on a distribution over θ, rather than just a point estimate, so that the bound is given by $\mathcal{L}(q)$ with $q(x, \theta)$ a function over both parameters. In this case x and θ are being treated identically, so the form of $\log p(\theta \mid y)$ in equation (2.17) is replaced with

$$\log p(\theta \mid y) \quad = \quad \mathcal{L}(q) + \mathrm{KL}(q \| p(x, \theta \mid y)). \tag{2.19}$$

where the lower bound is given by

$$\mathcal{L}(q) = \int q_x(x) q_\theta(\theta) \log \frac{p(X, Y, \theta)}{q_x(x) q_\theta(\theta)} dx d\theta. \tag{2.20}$$

The optimal choice for q is therefore $q(x, \theta) = p(x, \theta \mid y)$. However, no way of finding this will, in general, be available.

The problem can sometimes be made tractable by considering only q distributions from a restricted family and then finding the member of this family that maximizes the lower bound. There are multiple ways of restricting q, and any restriction leads to an approximation of the solution; lesser restrictions are likely to lead to solutions closer to the true one. This idea is the basis of approximate variational Bayesian methods. A common restriction on q is to consider the factorized family $q(x, \theta) = q_x(x) q_\theta(\theta)$ (complete factorization over all variables corresponds to the mean field approximation used in physics). Rearranging equation (2.20) in terms of one

of these factors, e.g. $q(\theta)$ gives

$$\mathcal{L}(q) = \int q_\theta(\theta) \left[\int q_x(x) \log p(x, y, \theta) dx \right] d\theta + \int q_\theta(\theta) \log q_\theta(\theta) + c, \quad (2.21)$$

where c is constant with respect to θ; a similar expression can be found with respect to $q(x)$. The term inside the square brackets can be recognized as $\mathbb{E}_{q(x)}(\log p(x, y, \theta)$, so the aim is to maximize this lower bound with respect to $q(\theta)$. It can be shown (by writing $\log \tilde{p}(y, \theta) = \mathbb{E}_{q(x)}(\log p(x, y, \theta) +$ const. and recognizing the result as a negative Kullback-Leibler divergence between $\tilde{p}$ and q_θ; see e.g. [39]) that this is minimized when

$$\log q_\theta(\theta) = \mathbb{E}_x \left(\log p(x, y, \theta) \right) + \text{constant w.r.t. } \theta,$$

and similarly for $q(x)$. The EM algorithm can be recovered as a special case of this, when $q_\theta(\theta) = \delta_{\{\theta\}}$, i.e. q is restricted to have a delta function θ marginal. In this case maximizing equation (2.21) with respect to $q_\theta(\theta)$ leads to the maximization over θ found in the M-step, equation (2.18).

As with EM methods, this general formulation does not specify the details of practical variational schemes, and these may be (and often are) difficult to derive. The problems involved are often only tractable for distributions in conjugate families, since in that case only finite sets of parameters need be updated [46]. However, variational schemes have been developed in some useful cases. In particular [46; 47] derive a variational Bayesian scheme for parameter estimation in the linear Gaussian case.

If variational Bayesian methods are computationally efficient but often difficult to derive and approximate, sampling methods are their opposite: often easy to derive and asymptotically exact, but computationally demanding. Like the EM algorithm, most sampling approaches, take advantage of the fact that for state space models it is often easier to estimate the joint posterior distribution $p(x_{1:T}, \theta \mid y_{1:T})$ over states and parameters than the posterior over parameters alone $p(\theta \mid y_{1:T})$. In doing this they make use of the fact that marginalization in sampling methods is trivially

easy: simply retaining the variables in each sample over which marginalization is required produces a sample based approximation of the marginal distribution.

There are a number of sampling-based approaches to parameter estimation. The simplest of these are MCMC schemes (see section 2.2) targeting the posterior $p(x_{1:T}, \theta \mid y_{1:T})$. These can usually be constructed via a Gibbs sampler that, for a parameter θ_i, targets the conditional $p(\theta_i \mid x_{1:T}, y_{1:T}, \theta_{-i})$, where θ_{-i} is the set of parameters excluding θ_i. It is often possible to find forms of this distribution that can be efficiently sampled, for example having standard distributions. However, if not and if the state transition and observation function can be evaluated, it is possible to use a Metropolis-within-Gibbs scheme (see section 2.2) by targeting

$$\begin{aligned} p(\theta_i \mid x_{1:T}, y_{1:T}, \theta_{-i}) &\propto p(y_{1:T} \mid x_{1:T}, \theta) p(x_{1:T} \mid \theta) p(\theta_i \mid \theta_{-i}) \\ &= p(\theta_i \mid \theta_{-i}) p(x_1 \mid \theta) \prod_{t=1}^{T} p(y_t \mid x_t, \theta) \prod_{t=2}^{T} p(x_{t+1} \mid x_t, \theta), \end{aligned}$$

where this latter equality is due to the model structure in figure 2.1. Evaluation of this density (the full joint likelihood) is expensive and is required for each sample of each parameter, so this 'sledgehammer' formulation should only be used as a last resort, but can be handy if all else fails.

As with state estimation, the Particle MCMC methods of [33] offer alternatives to MCMC for parameter estimation using a hybrid of sequential Monte Carlo and MCMC methods. In fact, [33] offers two different methods applicable to parameter estimation: the Particle Marginal Metropolis-Hastings (PMMH) sampler and the Particle Gibbs (PGibbs) sampler. These are described in detail in sections 4.3.3 and 4.3.4, respectively. The PMMH sampler allows samples to be drawn from $p(x, \theta \mid y)$, whereas the PGibbs sampler allows exact samples of $p(x \mid y, \theta)$ to be drawn using a particle filter, which is useful when efficient Gibbs samplers for $p(\theta \mid x, y)$ are available.

Online Parameter Estimation

If sequential parameter estimates are required, different methods must be employed. Filters making use of online parameter estimates are often called *adaptive filters* and these have been investigated since shortly after the development of the Kalman filter in the early 1960s. For linear Gaussian state space models, the problem consists of learning the state transition, observation and noise covariance matrices; [48] summarizes much of the early work in this area, dividing it into Bayesian [49], maximum likelihood, correlation of innovations [50], and covariance matching methods. The Bayesian methods in [48] give a general but computationally demanding recursive update for the distribution of parameters that is only really tractable in closed form for certain special cases, in particular noise covariance estimation, when conjugate priors can be used.

More recent Bayesian approaches have been based around multiple model (MM) methods, first introduced in [49]. These are extensively reviewed in [51] and are based on the assumption that the system is unknown but drawn from one of a number of alternative models. The filter is run for each of these models and their posterior probability is sequentially updated. There are multiple ways of combining the state estimates from this collection of models to get overall state estimates, but a truly Bayesian approach leads to a state estimate that is a mixture model over all models in the model set. The MM approach is based on the assumption that there is a single underlying model across all time periods but the idea has been extended to incorporate systems in which the model can change over time. These interacting multiple model (IMM) algorithms were first introduced by [52] and are also reviewed in detail in [51]. Transitions between models are governed by a finite state space Markov chain and thus the models are known *jump Markov (linear) models*. Recursive Bayesian estimation can be used to give a posterior distribution over models.

IMM methods can be applied to parameter estimation problems by defining a series of models with a range of parameter values arranged, for

example, in a grid over the parameter space. This has the obvious drawback of requiring a number of models exponential in the number of parameters to be estimated, so is only suitable for small parameter spaces. Furthermore, it is limited to a discrete set of parameter values and requires a transition model to be defined between parameter values, which may not be obvious. However, the IMM method has been applied to noise estimation in linear systems in e.g. [53] and [54] where, in the latter of these, it is used as a benchmark estimator and performs well.

The recent work of [54] introduces a sequential variational Bayesian approach to the adaptive filtering problem for linear Gaussian models through the careful use of a dynamic model for the parameters that maintains conjugate distributions for the noise covariance terms. [54] also draws a link between this method and earlier innovation correlation methods such as in [50]. It is suggested in [54] that this method can be extended to work with the EKF and UKF models.

Many of the methods for sequential parameter estimation in linear systems can be extended to non-linear cases using the EKF and UKF, at least in an informal way, e.g. [55], [56]. With non-linear filters it is also possible to include parameters in an augmented state space with their own dynamic models; this technique has been used in particle filtering [57], though not without criticism (see section 2.3.4). Much recent effort in non-linear sequential parameter estimation has been focussed on parameter estimation techniques for particle filters, and these are covered in section 2.3.4.

Likelihood-Free Parameter Estimation

Likelihood-free parameter estimation deals with the problem of parameter (and usually also state) estimation for systems for which it is impossible to evaluate the likelihood of the observed data. This can occur in situations in which the state or observation density cannot be directly evaluated, but where it is possible to simulate from these densities. In some cases, simulation might simply be much easier than evaluation of the density of interest

[58], but in certain special cases exact methods for the simulation of diffusions exist [59; 60] that allow exact sampling from the diffusion in cases where the transition density is only approximately tractable. Some of the methods already encountered can be used in certain situations in which it is not possible to calculate a likelihood. For example, the bootstrap particle filter [61] (see section 2.3), allows approximate estimation in cases in which evaluation of the state transition density impossible, but simulation is possible. Particle MCMC methods [33] based on this filter can be used in similar situations and allow exact (in an MCMC sense) samples to be drawn from the posterior of the parameters even when the transition density cannot be evaluated. These methods are used for parameter estimation in e.g. [58]. Both these methods rely on having tractable observation densities, so are not suitable in all likelihood-free situations (i.e. when the observation density is not tractable).

Another class of likelihood-free methods are *approximate Bayesian computing* (ABC) methods, first introduced in the field of population genetics by [62]. The most basic version of this idea is to draw a parameter sample θ from its prior $p(\theta)$, then to simulate the observations (or state in a directly observed system) using this set of parameters. If the resulting simulated observations are equal to the actual observations then θ is a sample from the posterior $p(\theta \mid y)$. This is a simple consequence of Bayes' rule $p(\theta \mid y) \propto p(y \mid \theta)p(\theta)$. In general, in continuous models this will happen with probability 0 and so the method relies on the approximation of accepting θ as a sample from the posterior when the simulated observations y_{sim} lie within some distance of the true observations y, i.e. if $\|y_{sim} - y\| < h$. This approximation can be shown to converge to the true posterior as $h \to 0$, although the error scales with $O(h^d)$, where d is the dimensionality of the observation space. This motivates a search for good low dimensional summary statistics for the observations, and automatic methods to find them e.g. [63]. There are several refinements to this method. For example, the basic rejection sampling method can be replaced by one

based on MCMC [64] in an algorithm similar to a pseudo-marginal MCMC scheme (see section 2.2.3). A recent review of ABC methods is given in [65].

2.2 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods were first introduced in [66], which presented the famous *Metropolis* algorithm, later extended to the non-symmetric case and given a more statistical focus in [67], leading to the *Metropolis-Hastings* algorithm. These are algorithms for drawing samples from a *target* distribution π by setting up a Markov chain with the target as its *stationary distribution*. They have become extremely popular in statistics, starting with [68] in the early 1990s, as methods of drawing samples from the usually intractable posterior distributions arising in Bayesian statistical approaches. The basic Metropolis-Hastings algorithm works by drawing a sample from a proposal distribution q and accepting this proposal as the next state of a Markov chain with an *acceptance probability* α given by

$$\alpha(x, y) = \min\left(1, \frac{\pi(y)q(x \mid y)}{\pi(x)q(y \mid x)}\right), \tag{2.22}$$

where x is the current state of the chain and y is the proposal. Following [69], this can be shown to work because the chain thus created has a *transition kernel* $p(x_{t+1} \mid x_t)$ given by

$$p(x_{t+1} \mid x_t) = \alpha(x_t, x_{t+1})q(x_{t+1} \mid x_t) + \delta_{\{x_{t+1}=x_t\}}\left(1 - \int \alpha(x_t, y)q(y \mid x_t)dy\right),$$

with the first term on the right hand side arising from accepting a proposal and the second term arising from the probability of rejecting any proposal. Taking equation (2.22) for $\alpha(x_t, x_{t+1})$ and $\alpha(x_{t+1}, x_t)$ and multiplying each by the corresponding denominator of the fraction inside the minimum gives

$$\pi(x_t)q(x_{t+1} \mid x_t)\alpha(x_t, x_{t+1}) = \pi(x_{t+1})q(x_t \mid x_{t+1})\alpha(x_{t+1}, x_t).$$

Combing this with the transition kernel gives rise to the *detailed balance equations*

$$\pi(x_t)p(x_{t+1} \mid x_t) = \pi(x_{t+1})p(x_t \mid x_{t+1}), \tag{2.23}$$

which can be seen by noting that the terms arising from rejection cancel because in that case $x_t = x_{t+1}$ and thus $\pi(x_t) = \pi(x_{t+1})$. Integration with respect to x_t then gives

$$\int \pi(x_t)p(x_{t+1} \mid x_t)dx_t = \pi(x_{t+1}),$$

meaning that x_{t+1} is distributed according to the target density π, *if* x_t is distributed according to $\pi(x_t)$. Thus, π is the stationary distribution of the chain, meaning that if for some t the state of the chain is distributed π then it will continue to be thereafter. It remains to show that the chain will ever reach that distribution. Fortunately, it can be shown that for a chain with a stationary distribution, the only further conditions required for the chain to *converge* to that distribution and for that distribution to be the limiting distribution, i.e for $\lim_{n\to\infty} \|P^n(x_t, \cdot) - \pi(\cdot)\| = 0$, where P^n is the distribution of the chain after n steps, are that the chain be *irreducible* and *aperiodic*; see [70] and [71]. Irreducibility requires that q must be such that all possible states can be reached. Aperiodicity requires that q cannot only return to certain of states with a period greater than 1. In general, these conditions on q, along with domain considerations that ensure that q can reach the whole of the support of π, are "very weak" [72] and are the only restrictions on its choice.

The purpose of MCMC methods is to draw a series of samples X_t that can be used to approximately evaluate the expected value of function f of a random variable with the target distribution π, using the approximation

$$\mathbb{E}_\pi(f(X)) \approx \bar{f}_N = \frac{1}{N}\sum_{t=M+1}^{M+N} f(X_t).$$

In fact, this is the only meaningful way to use MCMC samples: even evaluating the probability of a variable drawn from the target being in some region S is evaluating a counting function $f(x) = \mathbb{I}_{\{x \in S\}}$. It is therefore very important that this approximation holds, and this is dealt with by the ergodic property of Markov chains e.g. [72; 73] that states that this holds in probability for positive recurrent (for which the existence of a stationary distribution is a sufficient condition), aperiodic Markov chains, i.e. that

$$p(\bar{f}_N \to \mathbb{E}_\pi(f(X))) = 1.$$

The ergodic property is often referred to as meaning that the time average of a process path converges to the space (or ensemble) average. Ergodicity allows a law of large numbers and central limit theorem to be developed for these methods, which ensure the correct asymptotic behaviour and establish rates of convergence of samples to the true values, respectively [70].

2.2.1 Site-by-Site, Gibbs and Independence Sampling

Almost any proposal distribution will give a Markov chain with the target distribution as its stationary distribution, but the choice of proposal can affect the number of steps taken for the chain to converge to that distribution, as well as its *mixing* properties, that is, the time it takes to explore the support of the stationary distribution once that has been reached. For example, if almost every proposal is rejected, reaching and exploring the target is likely to be a slow task, with very high correlation between successive samples (as most will be the same as their predecessor).

In high dimensions, this problem can be particularly acute, since areas of high target probability can be very small, with only low probability of proposals hitting them. To alleviate this, a single component or subset of components of the state can be updated at each step instead of the entire state. This was, in fact, the original scheme proposed by [66]. The accept-

ance probability for a proposal y_i for this *site* is given by

$$\alpha(x_i, y_i \mid x_{-i}) = \min\left(1, \frac{\pi(y_i \mid x_{-i}) q_i(x_i \mid y_i, x_{-i})}{\pi(x_i \mid x_{-i}) q_i(y_i \mid x_i, x_{-i})}\right), \tag{2.24}$$

where x_i is the component (or block of components) being updated and x_{-i} is the remaining portion of the state (so that $x = x_i \cup x_{-i}$). The conditional target distribution $\pi(\cdot \mid x_{-i})$ is called the *full conditional* of the i^{th} component. The acceptance ratio in equation (2.24) can be derived from that in equation (2.22) by setting the proposal $q(y \mid x) = q_i(y_i \mid x)\delta_{\{y_{-i}=x_{-i}\}}$.

A special version of site-by-site updating is given by the Gibbs sampler, introduced by [74] and [68]. This uses the full conditional as the proposal density, i.e. $q(y_i \mid x_i, x_{-i}) = \pi(y_i \mid x_{-i})$, leading to complete cancellation of the fraction in equation 2.24 and thus the acceptance of all proposals. When the full conditionals are tractable and easy to sample, such schemes are popular [75] because they are often computationally efficient and do not require the design of a proposal.

The choice of which components to sample together is referred to as a *blocking scheme* and it can make a significant difference to the mixing rate of the chain. This is because a sampler drawing a block of n components is sampling in the n-dimensional hyperplane defined by the corresponding n axes in sample space; single site updates are sampling along the line of one of the state space axes. Thus, if the corresponding distribution in that hyperplane is relatively compact, only small moves will be likely, leading to slow convergence. For example, the rate of convergence of a Gibbs sampler for a bivariate normal can be shown to be given by the square of the correlation between components [73], with highly correlated distributions converging more slowly. This can be seen intuitively in figure 2.3. Thus, blocking together highly correlated components can improve efficiency [69].

A simple proposal method that can be used effectively inside site-by-site schemes is the *independence sampler*, first discussed by [67] and generalized by [70], in which the proposal function does not depend on the current

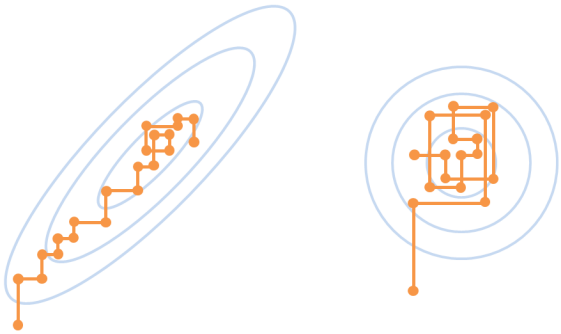


Figure 2.3: Strongly correlated components (left) can cause slow convergence for site-by-site schemes compared to uncorrelated components (right) because these schemes can only sample in the direction of one axis in each step

state, giving an acceptance probability of

$$\alpha(x, y) = \min\left(1, \frac{\pi(y)q(y)}{\pi(x)q(x)}\right).$$

In [76] this algorithm is compared to one-dimensional importance and rejection sampling and is found to work well when the proposal distribution is similar to the target distribution, but with heavier tails [69]. If the ratio $\inf_x \frac{q(x)}{\pi(x)} = 0$, then the algorithm is not geometrically convergent and thus can become stuck at certain points [73]. On the other hand, if this is not the case, then rejection sampling is possible, since a proposal distribution that can be scaled to be larger than π everywhere can be found (see e.g. [36]) and is probably preferable. In [73] it is therefore noted that "it is rare for the independence sampler to be useful as a stand-alone algorithm," but that "within a hybrid strategy which combines and mixes different MCMC methods, the method is extremely easy to implement and often very effective."

2.2.2 Adaptive MCMC

The proposal density and its relation to the target are crucial factors in determining the performance of MCMC schemes. However, before running

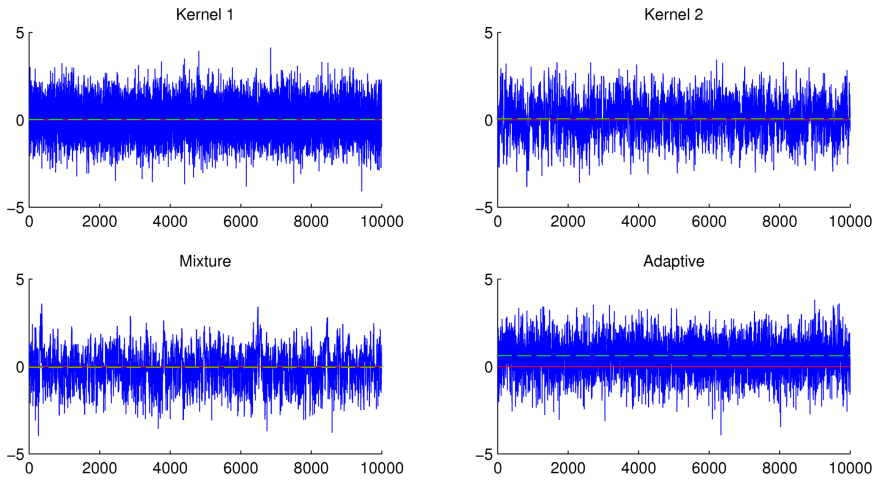


Figure 2.4: Example, from [78], of adaptive MCMC destroying ergodicity. Red lines show true mean, green dotted lines show sample mean for the chain shown. The adaptive scheme (bottom right) that does not target π is formed by using both kernel 1 and kernel 2 (top row) at various points, both of which individually target π. The mixture distribution formed using a mixture of kernels 1 and 2 (bottom left) does target the correct distribution π

an MCMC algorithm practically nothing might be known about the shape of the target distribution, so designing a proposal that produces 'good' results might be impossible. One possibility is to use a range of proposals with different properties to avoid the pathologies of any particular method [77]. Furthermore, as MCMC sampling progresses it will start to reveal information about the target distribution and the performance of the proposals. It is tempting, therefore, to use this information to refine the proposal, for example reducing its variance if rejection is too high.

These are examples of *adaptive* MCMC strategies, defined in the broadest sense as being MCMC strategies that use different transition kernels at each step. In general, however, such schemes must be used with great caution because they may not be ergodic with respect to π, even if all individual transition kernels target π. It should be noted that this is not the same as using *mixture model* proposals, which are single proposal densities and are therefore covered by the basic case. These are less desirable as

proposals because each component of the mixture has to be evaluated to evaluate the mixture density at a particular point. Non-convergence when using multiple kernels with the same stationary distribution is perhaps surprising but its effects can be seen in several simple but compelling counter-examples such as those in [78], [79] and [80]. Samples from the set-up in [78] are shown in figure 2.4. It is therefore important to understand which adaptive strategies *do* preserve π-ergodicity; this has been studied in particular by [81], [79] and [80].

The simple hybrid algorithm [77] with a kernel chosen independently of the current state at each step either randomly (*random sweep*) or in a cycle (*systematic sweep*) has been shown to converge correctly in [70]; [80] gives a straightforward proof. In fact, the individual kernels need not have π as a stationary distribution, as long as they can be combined to produce a single iteration of an MCMC sampler that does [77; 70]. This formulation can be seen to include the Gibbs sampler in which each kernel does not target π, since only some components are updated. However, when taken together, a full set of Gibbs update steps (i.e. updating every site) should target π. Some caution must still be exercised: [80] gives an example of a systematic sweep scheme that destroys irreducibility of the chain by making some states unreachable.

A related adaptive scheme allows the probability of choosing to update a particular component in a random sweep scheme to depend on the current state of the chain. In this case the acceptance probability must be modified in order to preserve π as the stationary distribution [69] from that in equation (2.24) to

$$\alpha(x_i, y_i \mid x_{-i}) = \min\left(1, \frac{\pi(y_i \mid x_{-i}) s(i \mid y_i, x_{-i}) q_i(x_i \mid y_i, x_{-i})}{\pi(x_i \mid x_{-i}) s(i \mid x_i, x_{-i}) q_i(y_i \mid x_i, x_{-i})}\right),$$

where $s(i \mid y_i, x_{-i})$ is the conditional probability of modifying site i.

Two other algorithms that allows continuous adaptation without damaging ergodicity are the adaptive direction sampling (ADS) and the related adaptive Metropolis sampling (AMS) algorithms of [82] and [72]. ADS

works by augmenting the state space to include a (fixed size) set of previous state points upon which adaptation is based. This is done by sampling along a line between the existing point and one of the previously visited states, with the idea that such a line is increasingly likely to traverse the high probability regions of the target. The algorithms were observed to have "inconsistent performance" on some problems on which traditional methods performed well. In general, algorithms that can incorporate the adaptive parameters into an extended state space retain ergodicity by using a single proposal on this extended state space.

General conditions on adaptation to preserve π-ergodicity have now been discovered [79; 80] and can be summarized broadly as precluding infinite adaptation in the kernel. More technically, they can be boiled down to the *diminishing adaptation* condition [80; 83] specified by

$$\lim_{n\to\infty} \sup_{x\in\mathcal{X}} \|P_{\Gamma_{n+1}}(x,\cdot) - P_{\Gamma_n}(x,\cdot)\| = 0 \quad \text{in probability}$$

where $P_{\Gamma_n}(x,\cdot)$ is the n^{th} transition kernel and $\mathcal{X}$ is the state space, and the technical *bounded convergence* condition

$$\{M_\epsilon(X_n,\Gamma_n)\}_{n=0}^{\infty} \quad \text{is bounded in probability,} \quad \epsilon > 0,$$
$$\text{for} \quad M_\epsilon(x,\gamma) = \inf\{n \geq 1 : \|P_\gamma^n(x,\cdot) - \pi(\cdot)\| \leq \epsilon\}.$$

[84] notes that this latter is "a technical condition that is satisfied for almost all reasonable adaptive schemes". An alternative approach is taken by [79], which also establishes ergodicity and some other results for chains with vanishing adaptation and shows that adaptive versions of the random walk Metropolis and independent Metropolis-Hastings algorithm can be constructed to satisfy the conditions given.

Since infinite adaptation is forbidden, a very simple solution is to stop adaptation after a certain fixed time period [78]. A more sophisticated adaptive MCMC and the first successful application of diminishing adaptation is the *adaptive Metropolis* algorithm of [85] and the corresponding ana-

lysis that showed that it retained ergodicity. This algorithm uses a Gaussian proposal with a covariance matrix that can adapt based on the entire state space history seen up to that point. The adaptation must be performed according to a specific formula that allows the ergodicity property of the resulting chain to be proven. [81] relaxes some of these conditions and gives an adaptive version of the standard random walk Metropolis algorithm.

The establishment of bounds on the adaptation process within which ergodicity is maintained has lead to numerous proposed adaptive schemes within those bounds. [86] proposed a single component version of the adaptive Metropolis algorithm. [83] propose two further schemes: state dependent scaling, which attempts to alter the scale of a Gaussian proposal to be optimal at the current point, and regionally adapted Metropolis, which divides the state space into a number of disjoint regions and then attempts to scale Gaussian proposals in each of these regions to be optimal. [86] proposes delayed rejection adaptive Metropolis in which, if a proposal is rejected, a number of refined proposals can be submitted, allowing local adaptation, while still retaining the desired stationary distribution. The adaptive version of this algorithm builds this delayed rejection strategy into an adaptive Metropolis algorithm, giving both local and global adaptability and an ability to counter poor calibration of either.

A different approach is taken by the *regeneration* algorithm of [87]. This is based on the idea of Markov chain regeneration which, in discrete state spaces, occurs when the chain revisits some nominated state, and which with some work can be extended to continuous state space Markov chains. At regeneration points the sample path of the Markov chain to the next regeneration is independent of the sample path from the previous regeneration point to the current one. This allows the transition kernel of the chain to be adapted at that time using the entire history of the chain up to that point as a basis for the adaptation. The algorithm allows an unlimited amount of adaptation but in high dimensional spaces is limited by the difficulty of obtaining frequent regeneration [87].

Adaptive MCMC continues to be an active area of both theoretical and applied research, for example [88] develops some further limit theorems for adaptive MCMC and [89] apply adaptive MCMC to image restoration.

Optimal Acceptance Rates

If adaptive methods can adjust their proposals to control the acceptance rate, it is natural to ask what acceptance rates they should target. There are some limited theoretical and experimental results on this. Results in [90] and [91] that show that for one-dimensional samplers with Gaussian proposals and targets, 0.44 is the optimal acceptance rate. It is also shown that under certain conditions the optimal acceptance rate for MCMC methods with Gaussian proposal and target is exactly 0.234 as the dimensionality of the system goes to infinity.

2.2.3 Variants of MCMC

Along with adaptive MCMC a wide variety of other methods exist for improving the efficiency and applicability of MCMC. This section briefly describes a few important variants, including methods to suppress wasteful random walk behaviour; slice sampling, an alternative sampling scheme to Metropolis-Hastings; ensemble methods for highly multimodal posteriors; reversible jump MCMC for variable dimensional state spaces; exact sampling methods; and pseudo-marginal methods that allow sampling using only unbiased likelihood estimates.

Suppressing Random Walk Behaviour

The standard Metropolis-Hastings algorithm is based around a random walk, with proposals made in random directions at each step. Even if every proposal is accepted, this still requires roughly n^2 steps in order to reach points n step-sizes away, due to the diffusion behaviour of random walks. This can lead to slow convergence and mixing, because it takes many steps

to reach the stationary distribution and subsequently to explore it. Particularly if each step is expensive to compute, it can be desirable to try to reduce this random walk behaviour by trying to encourage moves in useful directions. Several methods have been proposed to try to help with this, and such methods are worth investigation when standard MCMC methods exhibit slow exploration of the state space.

Overrelaxation [92], generalized to systems with non-Gaussian conditional distributions in [93], aims to suppress random walk behaviour in order to improve performance in highly correlated systems by drawing Gibbs samples that are negatively correlated with the previous samples in such a way as to leave the conditional distribution unchanged. For non-Gaussian systems [93] extended this method using order statistics: k proposals are drawn from the conditional, ordered along with the current sample, and then the one at the opposite position in the order to the current sample is selected (i.e. if the current sample is in i^{th} place, sample $k - i$ from the ordered list including the original sample is chosen). Again this can be shown to leave the conditional distributions unchanged, but can lead to dramatic speed-ups in simulation when the conditional densities are easy to sample. Clearly, this method is only useful when an ordering is possible on the samples and so, in general is only applicable to one-dimensional conditionals.

The *Hamiltonian* Monte-Carlo of [94] uses gradient information about the target in addition to the target function itself in order to direct the motion of the MCMC random walk. The method uses a state augmented with momentum variables and is, conceptually, related to the ADS scheme [82] discussed above, which also aims to use recent moves to choose the direction of the current sampling step. A step in the chain can either randomly update the momentum variables or use them to direct the current motion according to mechanical principles, with the target function acting as an energy function, hence its gradient being used to update the momentum from step to step.

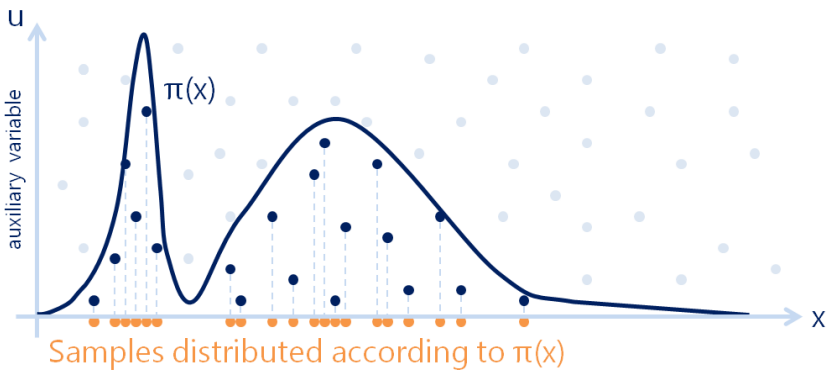


Figure 2.5: The intuition behind slice sampling is to uniformly sample in a space of one higher dimension than the target function π by introducing an auxiliary variable u. Only the non auxiliary part of the samples falling beneath the target are retained, but these are drawn from a probability density proportional to the target

Slice Sampling

Slice sampling [95; 96] is an alternative MCMC scheme to Metropolis-Hastings which is based on an efficient implementation of the intuition that the density of a function in an area can be approximately measured by counting the number of uniformly random samples drawn from an enclosing box that lie underneath it. The number of samples underneath the function in each area will be proportional to the density of the function there. Thus, by retaining only these samples, a set of samples with density proportional to that of the target function can be created. Figure 2.5 shows this idea.

[95; 96] set out an efficient, locally adaptive scheme for this, illustrated in figure 2.6. The idea (in one dimension) is roughly as follows. Starting at a current sample x^i, the auxiliary variable u' is drawn from a line bounded by the current target value $\pi(x')$. A *slice* is then generated as a line (hyperplane in the general case), with ends determined initially by being some width w away from the current sample x^i, but being allowed to move outwards until they are outside the target (i.e. until $u' > \pi(x_L)$ and $u' > \pi(x_R)$). The width w is a parameter of the method chosen by the user. A sample for x' is then drawn from this slice and, if it falls below the target i.e. if $u' < \pi(x')$ it is

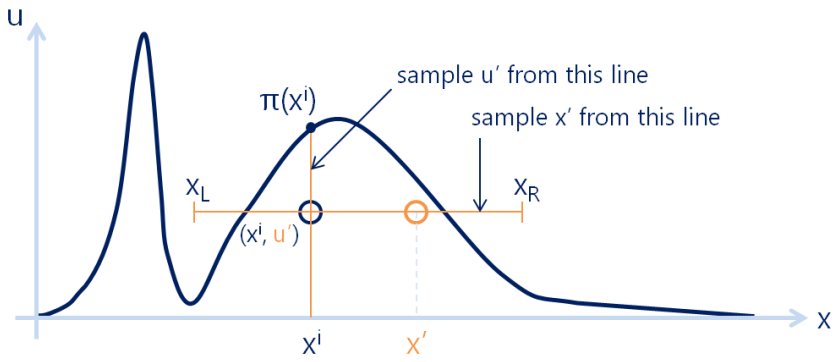


Figure 2.6: Slice sampling scheme from [96], starting at x^i and sampling x'

retained as a sample from π. Otherwise, the corresponding end point x_L or x_R to which it is closest is moved inwards and the process repeated. It is possible to show that this scheme satisfies detailed balance and thus is an MCMC scheme that targets π.

Slice sampling can be used in similar situations to Metropolis-Hastings, including in the update of individual components of the state from their full conditional densities. It offers an alternative scheme that, while it still contains a step-size parameter w, is somewhat less sensitive to its choice than Metropolis-Hastings is to the width of its proposal density. Like Metropolis-Hastings or Gibbs sampling, slice sampling can be used as a sampling component within larger MCMC schemes and merits investigation in cases when standard Metropolis-Hastings or Gibbs schemes exhibit poor behaviour. [96] shows how overrelaxation can be used with slice sampling in order to suppress random walk behaviour.

Simulated Tempering and Extended Ensemble MCMC

The convergence phase of MCMC can be thought of as being related to stochastic optimization, since it is generally aiming to find areas of high probability density in the target distribution. As with optimization, multimodal target distributions can cause difficulty for MCMC schemes, with the chain risking getting stuck for very long periods in local target max-

ima if the step size is insufficient to escape them. Hamiltonian and ADS type methods can accentuate this problem by encouraging the chain remain trapped in these areas. Therefore efforts have been made to incorporate heuristic techniques from multi-modal optimization such as simulated annealing [37] into MCMC methods to help improve mixing. Simulated annealing aims to allow better exploration of a multimodal target through the introduction of a series of related, but less 'peaky' distributions, often produced through exponentiation of the target by an exponent between 0 and 1. These can be thought of as corresponding to energy or temperature levels, with transitions throughout the distribution easier at high energies (exponents near 0).

A naive attempt to apply simulated annealing to MCMC might start with the highest energy distribution and progress to lower energy distributions over time, as in simulated annealing. When the lowest energy level (the target distribution) is reached, a random walk could be run from there with the aim of starting in an area of high target density. Such a scheme can, however, be shown to produce biased samples. [97] and [98] proposed a modified scheme, *simulated tempering*, which removes this bias, allowing simulated annealing ideas to be used within MCMC schemes by allowing the chain to make transitions between the various energy levels. Only samples coming from the lowest energy level, the true target distribution, are retained. The idea of this is that by having the ability to move into easier to explore higher energy levels, the chain will be able to to mix more effectively, accessing more of the target space more quickly.

A related idea is that of *extended ensemble* MCMC, including (similar or identical) methods known by the names *replica exchange Monte Carlo* [99], *Metropolis-coupled* MCMC [100] and *parallel tempering*. Unlike simulated tempering which uses a single chain that can transition between energy levels, these methods create an independent chain at each energy level, and introduce a transition kernel that can exchange the value of the chains at neighbouring energy levels, an idea first introduced in [99]. This can be

shown to preserve the stationary distributions at each energy level and allows high energy chains which should be able to explore the space easily to 'transmit' their new positions to lower energy chains. [101] reviews a large number of related methods in this area. The almost independent nature of the chains lends these methods to efficient parallel implementation, at least with parallelization up to the number of chains.

A tricky practical problem with the implementation of these methods is the choice of appropriate energy levels; [102] considers theoretical approaches to optimal scaling of these.

Exact Sampling

A different idea is that of *exact sampling* or *coupling from the past*, originally proposed by [103]. This is based around the idea of *coupled* Markov chains (ones that share a random number generator). The idea is that once two coupled chains get into the same state, since they share a random number generator, they will never separate in the future. This is called *coalescence*. If coupled chains were started from all possible initial states then, after they had coalesced, they must be in a state from the stationary distribution since any possible starting value would have arrived at that coalesced state. However, it is not sufficient just to simulate forward until coalescence occurs because the conditions under which coalescence occur might themselves affect the distribution (e.g. if points only coalesce when one state element reaches 100 then sampling after seeing coalescence will deliver biased samples).

To get around this, simulation is run to time 0 from some point in the past, $-T$. If by time 0 coalescence has not occurred then simulation is restarted from further back, e.g. $-2T$. Once the starting point is far back enough so that coalescence has occurred by time 0, the time 0 sample can be taken as an exact sample from the stationary distribution. However, there are usually far too many possible starting points (often an infinite amount) for simulation from them all to be feasible, so the method only works if

a small number of chains can be used to bound the entire ensemble. For example, if one dimensional chains never cross then simulations can be run from the most extreme starting points and when these chains have coalesced, it is certain that all possible chains would have done so. This algorithm is more easily applied to finite state spaces, but coalescence is related to the idea of regeneration used by [87] in continuous spaces. [104] links the idea of perfect sampling to regenerative versions of simulated tempering schemes and relaxes the backward-time idea described above so that forward-time simulation can be used.

Reversible Jump MCMC

In some problems the dimensionality of the state space can be unknown, for example in model selection problems where the number of parameters of the model is itself one of the unknown parameters. In this case the standard MCMC method must be adapted to allow a variable-dimensionality state space. This can be done through the use of *reversible jump* MCMC (RJMCMC), first introduced by [34].

RJMCMC formalizes the use of a proposal and its reverse that move between spaces of different dimensionality. It does this through the use of invertible mappings $h((x,u)) = (x',u')$, differentiable in each direction, (diffeomorphisms) that map from variables in one space X (so that $x \in X$) to another space of possible different dimensionality X' (so that $x' \in X'$). The u and u' variables are collections of the random variables necessary to generate the proposals, for example the standard Gaussian random variables in a random walk proposal. Thus $h : X \times U \to X' \times U'$ and $h^{-1} : X' \times U' \to X \times U$. These transforms replace the standard proposals in the acceptance ratio in equation (2.22), with the requirement for differentiability arising from the use of a change of variables in the integration of the detailed balance equation (2.23); see e.g. [34; 105] for further details.

In practice reversible transition proposals between spaces can be used

as the h transforms (see, for example, chapter 4), and these can be used in place of the standard proposal density and its reverse in equation (2.22). Thus RJMCMC is widely and easily applicable to many problems, particularly in model estimation.

Pseudo-Marginal MCMC

A recent introduction to the MCMC sampling field has been the Particle MCMC methods of [33]. These methods are covered in detail in chapter 4, but are essentially an application of the pseudo-marginal MCMC ideas introduced in [106] and extended and formalized in [107], using sequential Monte Carlo to obtain an unbiased approximation of the target density. Pseudo-marginal MCMC is based on using an unbiased Monte-Carlo approximation $r(x)$ of the target in place of the target $\pi(x)$ in the acceptance probability in equation (2.22). This can be seen to work (following [108]) by introducing a variable

$$w(x) = \frac{r(x)}{\pi(x)}, \tag{2.25}$$

which quantifies the Monte-Carlo error in $r(x)$. The variables x and w can be sampled using standard MCMC by introducing a target distribution $\tilde{p}(w, x) = p(w \mid x)r(x)$ over x and w using a proposal $q(x', w' \mid x, w) = p(w' \mid x')q(x \mid x')$. The conditional $p(w \mid x)$ is easy to sample through the relationship of w and x in equation (2.25).

The acceptance ratio of a chain targeting (x, w) is given by

$$\begin{aligned} \alpha((x', w'), (x, w)) &= \frac{\tilde{p}(x', w')p(w \mid x)q(x \mid x')}{\tilde{p}(x, w)p(w' \mid x')q(x' \mid x)} \\ &= \frac{r(x')q(x \mid x')}{r(x)q(x' \mid x)}, \end{aligned} \tag{2.26}$$

using the definitions of $\tilde{p}$ and $r(x)$. The marginal of the target of this chain

$\tilde{p}(x, w)$ can be written as

$$\begin{aligned}\tilde{p}(x) &= \int \tilde{p}(x, w)\mathrm{d}w \\ &= \pi(x)\int p(w \mid x)w(x)\mathrm{d}w \\ &= \pi(x)\mathbb{E}\left(w \mid x\right),\end{aligned}$$

where the relationship in equation (2.25) is used to go from the first to second line. So, if $\mathbb{E}(w \mid x) = c$, where c is constant, then the marginal $\tilde{p}(x)$ will be proportional to the target $\pi(x)$. For this to be the case it is necessary for

$$\begin{aligned}\mathbb{E}\left(w \mid x\right) &= \mathbb{E}\left(\frac{r(x)}{\pi(x)} \mid x\right) \\ &= \frac{1}{\pi(x)}\mathbb{E}\left(r(x) \mid x\right) = c,\end{aligned}$$

i.e. for $\mathbb{E}\left(r(x) \mid x\right) = c\pi(x)$. This is satisfied (with $c = 1$) when $r(x)$ is an unbiased estimator of $\pi(x)$ for all x. Therefore, this method allows unbiased estimators of a target to be used as in equation (2.26) to create an MCMC method with the true target as its stationary distribution. For state space models, particle filters provide such an unbiased estimator of likelihood and so can be used to draw samples from a target, as in Particle MCMC methods [33].

2.2.4 Diagnostics

The methods outlined so far in this section allow MCMC samples to be generated from the target distribution once the chain has converged to the stationary distribution. The question of how to tell when a chain has reached its stationary distribution still remains. Diagnostics of this must always be heuristic, since in general nothing is known about the target distribution and so any test of 'convergence' can only measure convergence between MCMC steps and not against the true distribution. There is always the risk, therefore, of misdiagnosis of slowly mixing chains as converged. The

problem is somewhat analogous to that of detecting the optimum in global optimization: it is easy to tell that a local optimum has been reached but essentially impossible (in reasonable time) to determine whether or not it is the global optimum.

Such problems "lead many theoreticians to conclude that all diagnostics are fundamentally unsound" [109]. Nevertheless, there have been a very large number of suggestions for ways in which MCMC convergence can be detected. [109] lists thirteen different convergence diagnostics from the literature each with their own pathological cases and successes. Further sets of convergence diagnostics are examined in [110] and [111]. Many diagnostics, such as that found in [69], are based on the statistical comparison of the output of multiple chains with different starting points, with convergence being assumed if they are sufficiently similar. Visual inspection also remains popular, with [112] and [113] providing some enhanced visualization methods for assessing convergence. A risk with any convergence criteria is that its use will introduce some systematic bias into the samples produced. This is examined for a number of convergence diagnostics in [114].

A recent general guide to MCMC convergence analysis is given in [115]. The conclusion of this and other studies is that MCMC diagnostics can be useful but should be used with caution since all have faults and to be most effective a number should be used together so as to minimize their chance of serious misdiagnosis.

2.3 Sequential Monte Carlo Methods (Particle Filters)

Sequential Monte Carlo (SMC) methods, often known as *particle filters* in the domain of state space models, are approximate methods able to sequentially sample from a sequence of distributions of increasing dimensionality. They are based on importance sampling as shown in equation (2.15) and can be thought of as approximating a distribution $\pi(x)$ of interest as a

weighted collection of samples (*particles*) x^i so that

$$\pi(x) \approx \sum_i w_i \delta_{\{x^i\}},$$

with weights given by each sample's importance weight $\frac{\pi(x^i)}{q(x^i)}$, where q is the importance distribution from which samples were drawn. Integration over this distribution can then be approximated as a sum, so that

$$\mathbb{E}_\pi(h(x)) = \int h(x)\pi(x)dx \approx \sum_i w_i h(x^i).$$

Unlike MCMC methods, they can be updated sequentially as more data becomes available, and so are of particular interest for on-line estimation. In state space models, they can be used to draw samples from successive filtering distributions for the full sequence of state variables $p(x_{1:t} \mid y_{1:t})$. SMC methods can also be used to sample from other distributions via the *SMC sampler* algorithm [116], which can be used in similar applications as MCMC methods. Choices of intermediate distributions similar to those in simulated tempering can allow highly multimodal distributions to be sampled effectively.

The idea of using importance samples in filtering evolved from earlier approximation schemes that used grid-based approximations either based on point masses on regular grids [117], or on spline approximations of various complexities, e.g. [118]. Limited computational resources in the 1970s meant work in that era tended to focus on more sophisticated interpolation schemes to reduce the storage burden of dense grids [119]. The early 1980s saw a hiatus of interest in such approximations, but by the late 1980s increases in computing power, particularly storage, "motivated a renewed look at simpler methods" [119]. [120] used piecewise linear approximations and [119] proposed piecewise constant approximations to the densities of interest. A general problem with such methods was that the construction of the grid, especially when the density functions involved are multimodal,

can be nontrivial [120; 119; 61]. Sequential use of importance sampling approximations first appeared in [61]. The name *particle filter* seems to have originated in [121], although individual samples were referred to as particles before this, for example in [122].

2.3.1 Basic Algorithm

Importance sampling can be used to approximate the integrals of functions over the filtering distribution as in equation (2.15) to give

$$\int h(x_{1:t})p(x_{1:t} \mid y_{1:t})dx_{1:t} \approx \sum_i \frac{p(x^i_{1:t} \mid y_{1:t})}{q(x^i_{1:t})} h(x^i_{1:t})$$

where $q(x_{1:t})$ is an importance density for the sample. This can be thought of as defining an approximate distribution for $p(x_{1:t} \mid y_{1:t})$ as

$$\begin{aligned} p(x_{1:t} \mid y_{1:t}) &\approx \sum_i \frac{p(x^i_{1:t} \mid y_{1:t})}{q(x^i_{1:t})} \delta_{\{x^i_{1:t}\}} \\ &= \sum_i w^i_t \delta_{\{x^i_{1:t}\}}, \end{aligned} \tag{2.27}$$

where w^i_t is an importance weight for sample i. [123] calls this the *empirical measure* for the filter. Sequential updating of this density is sufficient for filtering and can be arranged by careful choice of the importance distribution q to make it sequentially calculable, so that

$$q(x_{1:t}) = q_1(x_1 \mid y_1) \prod_{s=2}^{t} q_s(x_s \mid x_{1:s-1}, y_{1:s}).$$

Using this definition, q can be defined recursively as

$$q(x_{1:t}) = q(x_{1:t-1}) q_t(x_t \mid x_{1:t-1}, y_{1:t}),$$

meaning that a sample of $x_{1:t}$ can be drawn from q by augmenting an existing sample from $q(x_{1:t-1})$ via a new x_t sample from the q_t distribution, which can be calculated at time t. Using this importance density, the em-

pirical measure in equation (2.27) can be expressed recursively as

$$\begin{aligned} p(x_{1:t} \mid y_{1:t}) &\approx \sum_i \frac{p(y_t \mid x_t^i) p(x_t^i \mid x_{1:t-1}^i)}{p(y_t \mid y_{1:t-1}) q_t(x_t^i \mid x_{1:t-1}^i, y_{1:t})} w_{t-1}^i \delta_{\{x_{1:t}^i\}} \\ &= \frac{1}{p(y_t \mid y_{1:t-1})} \sum_i w_t^{i*} \delta_{\{x_{1:t}^i\}} \end{aligned}$$

where

$$w_t^{i*} = \frac{p(y_t \mid x_t^i) p(x_t^i \mid x_{1:t-1}^i)}{q_t(x_t^i \mid x_{1:t-1}^i, y_{1:t})} w_{t-1}^i. \tag{2.28}$$

This form is chosen because the observation likelihoods $p(y_t \mid y_{1:t-1})$ are generally intractable, but by defining the tractable unnormalized weight w_t^{i*} and noting that the empirical density must be a normalized probability density, normalized weights as in equation (2.27) can be calculated as

$$w_t^i = \frac{w_t^{i*}}{\sum_i w_t^{i*}}.$$

This sequential update of the weights in equation (2.28) is the basis of the sequential importance sampling (SIS) algorithm. Furthermore, the unnormalized weights can be used to estimate the observation likelihood, since the normalizing constant at each time is $p(y_t \mid y_{1:t-1})^{-1}$, which is approximated by $\left(\sum_i w_t^{i*}\right)^{-1}$, so that

$$p(y_{1:t}) \approx \prod_{s=1}^{t} \sum_i w_t^{i*}.$$

In fact (and crucially for the use of the particle filter inside pseudo-marginal MCMC type methods) this can be shown to be both a consistent [123] and unbiased estimator of the likelihood [124] (proposition 7.4.1); a simpler proof of this latter is given in [125].

Using the state transition density as the importance density in each time period, i.e. choosing $q_t(x_t^i \mid x_{1:t-1}^i, y_{1:t}) = p(x_t^i \mid x_{1:t-1}^i)$, yields a particularly simple form of the filter, known as the *bootstrap* filter [61]. In this case the

weight update in equation (2.28) simplifies to $w_t^{i*} = p(y_t \mid x_t^i) w_{t-1}^i$. This form has the advantage that it is not necessary to evaluate the state transition density at any point, merely to sample from it. On the other hand, the performance of this importance distribution can be very poor if the state transition model has very little noise, since this results in a densely concentrated importance function that might not align well with the true posterior filtering distribution. The bootstrap method appears to have been introduced independently several further times: in [122], as the condensation method of [126] and as the sequential imputations method of [127]

Sequential Importance Resampling

The idea of importance sampling is to distribute samples into 'important' areas in the sample space, i.e. areas in which the target distribution has high mass. The perfect importance distribution is the target distribution itself, in which case all samples will have equal weight. In general, the variance of the sample weights gives a good idea of the quality of the approximation and is closely related to the Monte Carlo variance of estimates of functions calculated using the samples (equal in the case when calculating $\mathbb{E}(h(x))$ with $h(x) = 1$, although in this case Monte Carlo methods are rather redundant). Thus, it is generally sensible to try to maintain low variance amongst the sample weights. In an attempt to quantify this [128; 76] introduced the concept of *effective sample size*, defined as

$$N_{ESS} = \left(\sum_i (w_t^i)^2 \right)^{-1}$$

which attempts to quantify the variance of the importance sampling estimator in terms of an estimator based on N_{ESS} samples from the target distribution.

Unfortunately for the SIS scheme, it has been shown that weight variance will increase (stochastically) over time [129] (based on a theorem from [128]) and that typically this will lead to an exponential increase in the vari-

ance of estimated quantities [123]. In practice this means that after some, usually small, number of filtering steps almost all the weight will be associated with a single sample and the effective sample size will be very close to 1. This sample need not be a good approximation of the filter density, it is simply the best of a potentially very bad bunch.

To combat particle degeneration, the filtering distribution can be *resampled*, a process that aims to produce a new set of samples approximating the filter distribution but with a more even weight distribution. This new set of samples should maintain the property that the estimates of integrals of functions over it are unbiased. A resampled distribution can be given by

$$p(x_{1:t} \mid y_{1:t}) \approx \sum_i \frac{N_i}{N} \delta_{\{x^i_{1:t}\}},$$

with $\mathbb{E}\left(N_i \mid w_t^{1:N}\right) = N w_t^i$ sufficient to ensure that this new approximation is unbiased. Here N_i can be thought of as the number of *offspring* of sample i in the new approximation. It should be noted that the Monte Carlo variance of estimates based on this new approximation cannot be better than that of those based on the old approximation, so the latter should always be used for making these estimates [121; 130].

There are several popular approaches to resampling, the simplest being multinomial sampling, which simply involves drawing with replacement from the current set of samples, using their weights as their selection probability. Other popular approaches that aim to reduce Monte Carlo variance include residual resampling, which uses $N_i = \lfloor N w_t^i \rfloor + \mathbb{I}_{\{u < N w_t^i - \lfloor N w_t^i \rfloor\}}$ where $u \sim \mathcal{U}(0,1)$, stratified sampling, and systematic sampling [122]. These latter two approaches aim to sample the existing samples more evenly by arranging them by weight and sampling from each of N partitions. These methods are compared in [131], which finds stratified and systematic sampling methods perform best, followed by residual and then multinomial resampling; [121] note that stratified sampling will reduce Monte Carlo variance, although [131] found that these methods were not always

superior in practice.

Algorithm 1 Sequential Importance Resampling (SIR) filter

Initialize N particles: $x^i \sim q_1(x_1)$, $w_1^{i*} = \frac{p(x_1^i)p(y_1|x_1^i)}{q_1(x_1^i)}$

Normalize weights: $w_1^i = \frac{w_1^{i*}}{\sum_j w_1^{j*}}$

for $t = 2$ to T **do**

  Resample (if required):

    Draw N samples from resampling scheme and weight accordingly

  Propagate samples:

  for $i = 1$ to N **do**

    Sample: $x_t^i \sim q_t(x_t \mid x_{t-1}^i)$

    Weight: $w_t^{i*} = \frac{p(x_t^i|x_{t-1}^i)p(y_t|x_t^i)}{q_t(x_t^i|x_{t-1}^i)}$

  end for

  Normalize weights: $w_t^i = \frac{w_t^{i*}}{\sum_j w_t^{j*}}$

end for

Adding a resampling step to the SIS algorithm gives the *sequential importance* re*sampling* (SIR) algorithm that is the most common form of the particle filter in use. This algorithm can be summarized as shown in algorithm 1 [129]. Resampling can be performed every step or only when a measure of sample diversity such as ESS falls below a certain threshold [127]; this latter approach is probably preferable as it does not result in unnecessary loss of sample diversity due to resampling when not required.

Resampling causes some samples to be duplicated whilst others disappear and this inevitably causes a loss of diversity in early sample periods due to many samples sharing common ancestry; figure 2.7 illustrates this. This means that though they do approximate the distribution $p(x_{1:t} \mid y_{1:t})$, approximations of the distribution of x become increasingly poor with increasing lag. These distributions can be found through the use of smoothing methods as described in section 2.3.3.

Another drawback of resampling is that the samples produced are no longer independent and so analysis of the properties of the approximation is more complicated as classic limit theorems relying on independence of samples no longer apply [132]. However, several important results have

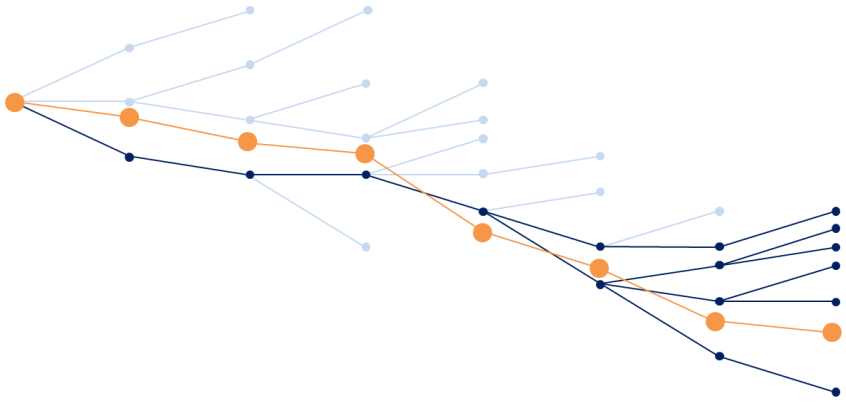


Figure 2.7: Illustration of the degeneracy of particle filter ancestry. Dark paths show ancestry of all current particles. These degenerate to a single common ancestor path at long lags. Orange path is true state path

been established for these methods, perhaps most importantly that they give consistent estimators of the distributions of interest. [133] gives what claims to be the first mathematically well-founded convergence results for interacting particle approximations, establishing that

$$\lim_{N\to\infty} \mathbb{E}\left(\left|\frac{1}{N}\sum_{i} f(x_t^i) - \pi_t(f)\right|\right) = 0,$$

where π_t is the target distribution at time t and f is a test function over the samples. [132] reviews a number of further convergence results for interacting particle systems, along with the conditions under which they hold.

Central limit theorems provide a means to establish not only the eventual convergence of the algorithms, but also to say something about the rate at which they do so. [134] gives the first central limit theorem for the paths of interacting particle systems and further related results are given in [124]. [135] gives a central limit theorem for sequential Monte-Carlo methods that applies to a large class of methods, including those using MCMC steps, "under minimal assumptions on the distributions" and looks at the asymptotic long-term stability of the algorithms.

There have also been a number of empirical studies of the performance of particle filter methods, such as [136] and [137], though these inevitably tend to be based around specific models.

2.3.2 Variants

Particle filters have been very popular methods for non-linear filtering and many refinements have been proposed to the basic algorithm, including many for specific applications. This section will attempt to present a few of the most significant and generally applicable variants in something of a unified context.

The main areas of design latitude in the particle filter are the choice of proposal function and the resampling method. In fact these can both be seen as modifying the overall importance density, since

$$q(x_{1:t} \mid y_{1:t}) = q_t(x_t \mid x_{1:t-1}, y_{1:t}) q_{1:t-1}(x_{1:t-1} \mid y_{1:t-1}), \tag{2.29}$$

where, in the basic filter with multinomial resampling, $q_{1:t-1}(x_{1:t-1} \mid y_{1:t-1}) = \sum_i w_t^i \delta_{\{x_t^i\}}$, since each new sample at t chooses an ancestor for earlier periods. However, it was established in the previous section that the optimal importance density is

$$p(x_{1:t} \mid y_{1:t}) = p(x_t \mid x_{1:t-1}, y_{1:t}) p(x_{1:t-1} \mid y_{1:t}), \tag{2.30}$$

so it is conceivable that schemes could be designed for both q_t and $q_{1:t-1}$ that better approximate this despite its general intractability. Schemes dealing with q_t are known as *adapted* (or approximately adapted) schemes and attempt to adapt the proposal at each step to the latest observation using either approximate schemes or MCMC kernels. Schemes dealing with $q_{1:t-1}$ include the *auxiliary* particle filter, a biased resampling scheme, and the *resample-move* scheme based on MCMC kernel moves and these attempt to better choose the sample of earlier $x_{1:t-1}$ states.

Adapted and Approximately Adapted Proposals

According to a comparison of equations (2.29) and (2.30), the optimal proposal density for the next state q_t is given when $q_t(x_t \mid x_{1:t-1}, y_{1:t}) = p(x_t \mid x_{1:t-1}, y_{1:t})$. In general, this density will be intractable; it also leads to a simple but often intractable form for the weight update in equation (2.28) $w_t^{i*} = p(y_t \mid x_{1:t-1}) w_{t-1}^i$. In the special case when the state transition model is (discrete-time) non-linear Guassian with linear Guassian observations the proposal density and the weight update can be calculated analytically [129].

In other cases, the idea of the adapted proposal can still be useful as a guide for proposal design, leading to approximately adapted proposals, which can be produced using a number of approximation methods. In particular [129] suggests local linearization of the state space model similar to the approximation used in the EKF and [138] suggests the use of the unscented transform as in the UKF to approximate the adapted proposal. This latter method has been especially popular due to its ease of application and effectiveness. In these cases, the standard weight update must be used, but as long as the approximation density is tractable this is straightforward.

Auxiliary Particle Filter

The key insight behind the auxiliary particle filter, introduced in [139], is that, in the optimal importance density above $q_{1:t-1}(x_{1:t-1} \mid y_{1:t-1}) = p(x_{1:t-1} \mid y_{1:t})$. Clearly $p(x_{1:t-1} \mid y_{1:t})$ cannot be approximated before the observation y_t is available, so the distribution of the original particles $x_{1:t-1}^i$ cannot be optimal. Auxiliary particle filtering attempts to approximate this distribution at time t and use it as the distribution from which to (re)sample $x_{1:t}^i$ at time t. The approximation is based on the observation

$$\begin{aligned} p(x_{1:t-1} \mid y_{1:t}) &\propto p(x_{1:t-1} \mid y_{1:t-1}) p(y_{t+1} \mid x_{1:t-1}, y_{1:t-1}) \\ &\approx \sum_i w_{t-1}^i \delta_{\{x_{1:t-1}^i\}} \int p(y_t \mid x_t) p(x_t \mid x_{1:t-1}^i) dx_t, \end{aligned}$$

where the first term is approximated by the particle approximation at $t-1$. In general the integral is not tractable, but if it can be approximated this can still provide a useful importance distribution for $x^i_{1:t-1}$. The idea presented in [139] is to approximate

$$p(x_t \mid x^i_{1:t-1}) \approx \delta_{\{\mu^i_t\}},$$

where μ^i_t is some easy to calculate point approximation of x_t given the preceding states, for example the mean of the state transition density so that $\mu^i_t = \mathbb{E}_{p(x_t|x_{t-1})}(x_t \mid x_{t-1})$. Then the integral is approximated as $p(y_t \mid \mu^i_t)$, so that

$$p(x_{1:t-1} \mid y_{1:t}) \;\tilde{\propto}\; \sum_i w^i_{t-1}\delta_{\{x^i_{1:t-1}\}}p(y_t \mid \mu^i_t),$$

where $\tilde{\propto}$ means "approximately proportional". This can be sampled by drawing ancestors with probability

$$u^i_t = \frac{w^i_{t-1}p(y_t \mid \mu^i_t)}{\sum_j w^j_{t-1}p(y_t \mid \mu^j_t)}.$$

A collection of ancestors sampled with these probabilities is no longer an unbiased approximation of $p(x_{1:t-1} \mid y_{1:t-1})$, but can be made so by weighting each particle by w^i_{t-1}/u^i_t. This can be seen as another application of importance sampling, since the unbiased estimator

$$\begin{aligned}\hat{p}(x_{1:t-1} \mid y_{1:t-1}) &= \sum_i w^i_{t-1}\delta_{\{x^i_{1:t-1}\}} \\ &= \sum_i \frac{w^i_{t-1}}{u^i_t}u^i_t\delta_{\{x^i_{1:t-1}\}}\end{aligned}$$

can be sampled by drawing weighted samples with $\sum_i u^i_t\delta_{\{x^i_{1:t-1}\}}$ as an importance density (which by construction is defined on exactly the support of the target $\sum_i w^i_{t-1}\delta_{\{x^i_{1:t-1}\}}$). The auxiliary particle filter algorithm is set out in algorithm 2.

[137] compare the auxiliary and bootstrap filters using a time-varying autoregressive test model and conclude that "the auxiliary particle filter gives a slight but systematic improvement in performance".

Algorithm 2 Auxiliary Particle Filter

Initialize N particles: $x^i \sim q_1(x_1)$, $w_1^{i*} = \frac{p(x_1^i)p(y_1|x_1^i)}{q_1(x_1^i)}$
Normalize weights: $w_1^i = \frac{w_1^{i*}}{\sum_j w_1^{j*}}$
for $t = 2$ to T **do**
  Calculate propagation weights:
  for $i = 1$ to N **do**
    Calculate μ_t^i (approximate summary statistic)
    Calculate propagation weight: $u_t^{i*} = w_{t-1}^i p(y_t \mid \mu_t^i)$
  end for
  Normalize propagation weights: $u_t^i = \frac{u_t^{i*}}{\sum_j u_t^{j*}}$
  Select N particles to propagate according to weights u_t^i
  Propagate samples:
  for $i = 1$ to N **do**
    Sample: $x_t^i \sim q_t(x_t \mid x_{t-1}^i)$
    Weight: $w_t^{i*} = \frac{p(x_t^i|x_{t-1}^i)p(y_t|x_t^i)}{u_t^{a(i)} q_t(x_t^i|x_{t-1}^i)}$ where $a(i)$ is the ancestor index of i
  end for
  Normalize weights: $w_t^i = \frac{w_t^{i*}}{\sum_j w_t^{j*}}$
end for

MCMC Methods: Resample-Move and MCMC Based Particle Filters

The resample-move algorithm of [140] introduced the idea of using MCMC kernels within particle filtering methods in order to 'rejuvenate' the particle collection. MCMC rejuvenation can be applied at any point during the particle filtering process to resample the current particle distribution from the true distribution $p(x_{1:t} \mid y_{1:t})$ by using an MCMC kernel that targets this distribution. The current particle positions are used as the starting values of the MCMC chain with the intuition that these particles are 'close' to being samples from the target distribution. Sampling the entire distribution $p(x_{1:t} \mid y_{1:t})$ will take at least $O(t)$ time, so cannot be used as part of

a sequential update step. Therefore, resample-move algorithms resample the the $x_{t-L:t}$ variables, for some fixed lag L. This makes them particularly successful in fixed-lag smoothing applications [130]. MCMC resampling is done by noting that

$$\begin{aligned} p(x_{1:t} \mid y_{1:t}) &= \prod_{s=t-L}^{t} p(y_s \mid x_s) \prod_{s=t-L}^{t} p(x_s \mid x_{s-1}) p(x_{1:t-L-1} \mid y_{1:t-L-1}) \\ &\approx \prod_{s=t-L}^{t} p(y_s \mid x_s) \prod_{s=t-L+1}^{t} p(x_s \mid x_{s-1}) \sum_i \frac{1}{N} p(x_L \mid x_{L-1}^i) \delta_{\{x_{1:t-L-1}^i\}}, \end{aligned}$$

using the sample-based approximation of $p(x_{1:t-L-1} \mid y_{1:t-L-1})$. This can be resampled by selecting a sample $x_{1:t-L-1}^i$ from the sample-based approximation of $p(x_{1:t-L-1} \mid y_{1:t-L-1})$, and then sampling the variables $x_{t-L:t}$ by running a Markov chain with acceptance probability

$$\alpha(x_{t-L:t}^{(k)}, x'_{t-L:t}) = \frac{\prod_{s=t-L}^{t} p(y_s \mid x'_s) \prod_{s=t-L+1}^{t} p(x'_s \mid x'_{s-1}) p(x'_L \mid x_{L-1}^i)}{\prod_{s=t-L}^{t} p(y_s \mid x_s^{(k)}) \prod_{s=t-L+1}^{t} p(x_s^{(k)} \mid x_{s-1}^{(k)}) p(x_L^{(k)} \mid x_{L-1}^i)},$$

where $x_t^{(k)}$ is the current state of the chain. A similar approach is taken in [141], where successive MCMC fixed-lag approximations are represented approximately using a sample-based representation and updated sequentially.

The resample-move algorithm is usually presented as a standard SIR filter with the addition of a move step after resampling. In this presentation, the algorithm can be seen as simply producing a refined proposal distribution $q_{1:t-1}$ for the particle ancestors at the next stage, since it uses an ancestor proposal of the form

$$q_{1:t-1}(x_{1:t-1}^i \mid y_{1:t}) \approx p(x_{1:t-1}^i \mid y_{1:t-1})$$

with the approximation here being provided by MCMC samples from (approximately) the target distribution $p(x_{1:t-1} \mid y_{1:t-1})$. Though an MCMC step could be used to target the more optimal $p(x_{1:t-1}^i \mid y_{1:t})$ the resulting

collection would, as with the auxiliary particle filter, be a biased approximation of $p(x^i_{1:t-1} \mid y_{1:t-1})$ and so to use this particle collection in the sequential update it would need to be weighted to correct for this. However, the weighting required is $\frac{p(y_t|x_{1:t-1})}{p(y_t|y_{1:t-1})}$; though the denominator can be found through normalization, the numerator may still be intractable (although not in, for example, the non-linear state transition, linear observation case, so this idea might be useful there).

MCMC resample-move was not the first algorithm to attempt to restore particle diversity by moving particles. It was preceded by kernel density type methods e.g. [61], which resampled from a kernel density approximation of $p(x_{1:t} \mid y_{1:t})$ rather than the point mass distribution coming from the importance sampling approximation. This effectively adds random noise to the samples as they are resampled, but choice of the kernel width can be tricky, especially in high-dimensional spaces.

While the resample-move algorithm presented in [140] uses a standard proposal step to generate new state samples, the MCMC-based particle filter method of [142] uses an MCMC step to generate samples from the next target distribution $p(x_t \mid y_{1:t})$. This is no longer a sequential importance sampling scheme, but an approximate sequential MCMC scheme that relies on a sample-based approximation of the preceding filter distribution and also on approximating the marginal $p(x_t \mid y_{1:t})$ as

$$\begin{aligned} p(x_t \mid y_{1:t}) &= \int p(x_t \mid x_{t-1}, y_t) p(x_{t-1} \mid y_{1:t}) dx_{t-1} \\ &\approx \int p(x_t \mid x_{t-1}, y_t) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \\ &\approx \frac{1}{N} \sum_i p(x_t \mid x^i_{t-1}, y_t) \\ &\propto \sum_i p(y_t \mid x_t) p(x_t \mid x^i_{t-1}), \end{aligned}$$

allowing approximate samples to be drawn recursively from $p(x_t \mid y_{1:t})$ via an MCMC chain set up to target the distribution $p(y_t \mid x_t) p(x_t \mid x^i_{t-1})$ for each sample x^i_{t-1}, i.e. making a proposal $x'_t \sim Q(x'_t \mid x^{(k)}_t)$ where $x^{(k)}_t$ is the

current sample, and accepting it with probability

$$\alpha(x_t^{(k)}, x_t') = \frac{p(y_t \mid x_t')p(x_t' \mid x_{t-1}^i)Q(x_t^{(k)} \mid x_t')}{p(y_t \mid x_t^{(k)})p(x_t^{(k)} \mid x_{t-1}^i)Q(x_t' \mid x_t^{(k)})}.$$

[142] recommends starting the chain with a sample from the transition model $x_t^{(1)} \sim p(x_t \mid x_{t-1}^i)$.

The need for the first approximation in the above method $p(x_{t-1} \mid y_{1:t}) \approx p(x_{t-1} \mid y_{1:t-1})$ can be obviated by sampling the joint density $p(x_t, x_{t-1} \mid y_{1:t})$ rather than the marginal, since

$$p(x_t, x_{t-1} \mid y_{1:t}) \propto p(y_t \mid x_t)p(x_t \mid x_{t-1})p(x_{t-1} \mid y_{1:t-1})$$

so, given a sample-based approximation of the previous filter distribution $p(x_{t-1} \mid y_{1:t-1})$ this can be sampled by choosing a sample x_{t-1}^i from the previous filter approximation and proposing $(x_t, x_{t-1})' \sim Q((x_t, x_{t-1})' \mid (x_t, x_{t-1})^{(k)})$. This proposal is then accepted with probability

$$\alpha = \frac{p(y_t \mid x_t')p(x_t' \mid x_{t-1}')Q\left((x_t, x_{t-1})^{(k)} \mid (x_t, x_{t-1})'\right)}{p(y_t \mid x_t^{(k)})p(x_t^{(k)} \mid x_{t-1}^{(k)})Q\left((x_t, x_{t-1})' \mid (x_t, x_{t-1})^{(k)}\right)}.$$

This is the insight behind the MCMC-Particles algorithm of [136].

An interesting modification of the MCMC based particle filter algorithm, also proposed in [142], is the use of RJMCMC proposal steps to allow the dimensionality of the state x_t to change with time (used there to account for a variable number of tracked objects); a similar idea could also be used in the MCMC-Particles algorithm.

Rao-Blackwellization

Rao-Blackwellization in the context of particle filtering is the idea of marginalizing part of the state because it has a tractable conditional distribution when conditioned on the remaining parts of the state, i.e. part of the state is conditionally linear Gaussian or of finite state. Denoting the condition-

ally tractable states x_L and the intractable states x_N, this amounts to the factorization

$$p(x_{1:t} \mid y_{1:t}) = p(x_{L,1:t} \mid x_{N,1:t}, y_{1:t}) p(x_{N,1:t} \mid y_{1:t}) \tag{2.31}$$

$$\approx \sum_i p(x_{L,1:t} \mid x^i_{N,1:t}, y_{1:t}) w^i_t \delta_{\{x^i_{N,1:t}\}}, \tag{2.32}$$

where the second line shows the importance sampling approximation. This latter approximation makes it apparent that, if $p(x_{L,1:t} \mid x^i_{N,1:t}, y_{1:t})$ is available in closed form, the overall approximate distribution will be a mixture distribution. In the case of x_L being conditionally linear Gaussian, the distribution $p(x_{L,1:t} \mid x^i_{N,1:t}, y_{1:t})$ can be found using the Kalman filter and the distribution in equation (2.32) will be a Gaussian mixture. This approach is sometimes referred to as the *mixture Kalman filter*. Rao-Blackwellization can be shown to reduce the variance of the particle weights [143] and often allows systems that would otherwise be computationally intractable to be tackled with particle filters, e.g. [144].

2.3.3 Smoothing

As already noted, the particle filter as presented in the preceding sections produces samples from the smoothing distribution $p(x_{1:T} \mid y_{1:T})$ as part of its operation (this is not strictly necessary, since for filtering in Markov models only the most recent marginal distribution need be stored at each step). However, the estimates produced in this way, called *filter-smoother* estimates in [145], become increasingly poor at increasing lags due to the degeneracy of particle paths in early periods, as illustrated in figure 2.7. Because of this degeneracy the filter-smoother is a very inefficient estimator of the smoothing distribution. On the other hand, it can be calculated in linear time and produces samples from the joint smoother distribution rather than from just the marginal smoothing distributions $p(x_t \mid y_{1:T})$.

As with other methods, smoothing algorithms can be developed in two broad flavours: forward-filtering backward-smoothing and two-filter ap-

proaches. Perhaps the simplest method is the forward-filtering backward smoothing method in [129] that allows samples to be drawn from the marginal smoothing distributions using a particle approximation to equation (2.10):

$$\begin{aligned} p(x_t \mid y_{1:T}) &= p(x_t \mid y_{1:t}) \int \frac{p(x_{t+1} \mid x_t)}{p(x_{t+1} \mid y_{1:t})} p(x_{t+1} \mid y_{1:T}) dx_{t+1} \\ &\approx p(x_t \mid y_{1:t}) \sum_j \frac{w_{t+1|T}^j p(x_{t+1}^j \mid x_t)}{p(x_{t+1}^j \mid y_{1:t})} \\ &\approx \sum_i w_t^i \delta_{\{x_t^i\}} \sum_j \frac{w_{t+1|T}^j p(x_{t+1}^j \mid x_t^i)}{\sum_k w_t^k p(x_{t+1}^j \mid x_t^k)} \\ &= \sum_i w_{t|T}^i \delta_{\{x_t^i\}}, \end{aligned}$$

where

$$w_{t|T}^i = w_t^i \sum_j \frac{w_{t+1|T}^j p(x_{t+1}^j \mid x_t^i)}{\sum_k w_t^k p(x_{t+1}^j \mid x_t^k)}.$$

The first approximation rests on a particle approximation of the integral and the second approximation rests on particle approximations of $p(x_t \mid y_{1:t})$ and $p(x_{t+1}^j \mid y_{1:t}) = \int p(x_{t+1}^j \mid x_t) p(x_t \mid y_{1:t}) dx_t$. This algorithm requires calculation of $p(x_{t+1}^j \mid x_t^i)$ for all i, j and thus has time complexity of $O(TN^2)$. The initial smoothing weights are given by the final filter weights, i.e. $w_{T|T}^i = w_T^i$. This algorithm results in a particle based smoothing distribution constructed using re-weighted versions of the original forward filter samples at each time period.

The forward-filtering backward-*sampling* method of [146] allows samples to be drawn from the approximate joint state distribution at a cost of $O(TN)$ per sample. The first (time T) sample X_T is drawn from the final filter distribution $X_T \sim p(x_T \mid y_{1:T})$. Subsequent samples, working back in time to

$t = 1$, can be drawn from

$$\begin{aligned} p(x_t \mid X_{t+1}, y_{1:T}) &= p(x_t \mid y_{1:t}, X_{t+1}) \\ &\propto p(X_{t+1} \mid x_t) p(x_t \mid y_{1:t}) \\ &\approx \sum_i w_t^i p(X_{t+1} \mid x_t^i) \delta_{\{x_t^i\}}, \end{aligned}$$

which can be sampled by drawing $X_t = x_t^i$ with probability $\frac{w_t^i p(X_{t+1}|x_t^i)}{\sum_j w_t^j p(X_{t+1}|x_t^j)}$.

It is also possible to draw an approximate MAP path from the particle filter, and this can be done forward in time, by treating the particle filter samples as a randomly placed but finite set of states at each time period. In this case, the samples can be treated like a finite state space grid of states and the Viterbi algorithm given in section 2.1.1 can be used to find the MAP path at an $O(N^2)$ time cost in each period [147].

The two-filter approach to smoothing in particle filters is complicated by the fact that the distribution $p(y_{t+1:T} \mid x_t)$ that must be targeted by the backward information filter is *not* a probability distribution with respect to x_t and need not even be finite [21; 15]. Early versions of the two-filter particle smoother [122] ignored this possibility. A generalized version of the two-filter smoother was introduced in [21] that deals with this problem by instead targeting an artificial backward distribution $\tilde{p}(x_t \mid y_{t:T})$ in the backward filter that *is* a probability distribution with respect to x_t. This is done by introducing an artificial prior $\gamma_t(x_t)$ so that

$$\tilde{p}(x_t \mid y_{t:T}) = \frac{p(y_{t:T} \mid x_t)\gamma_t(x_t)}{p(y_{t:T})}.$$

Any prior $\gamma_t(x_t)$ can be used, although it should have the same support as the true prior $p(x_t)$ (which in the general non-linear case will not be tractable) and must be able to be calculated exactly; [123] recommend using a heavy tailed approximation of the true prior $p(x_t)$, which can be defined recursively by defining $\gamma_1(x_1)$ and $\gamma_t(x_t \mid x_{t-1})$. The two filter decomposition is given in equation (2.3), and the particle approximation can be

constructed as

$$\begin{aligned} p(x_t \mid y_{1:T}) &\propto p(x_t \mid y_{1:t-1}) p(y_{t:T} \mid x_t) \\ &\propto \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \frac{\tilde{p}(x_t \mid y_{t:T})}{\gamma_t(x_t)} \quad (2.33) \\ &\approx \sum_i w_{t-1}^i p(x_t \mid x_{t-1}^i) \sum_j \frac{\tilde{w}_t^j}{\gamma_t(\tilde{x}_t^j)} \delta_{\{\tilde{x}_t^j\}} \\ &= \sum_j w_{t|T}^j \delta_{\{\tilde{x}_t^j\}} \end{aligned}$$

where

$$w_{t|T}^j \propto \frac{\tilde{w}_t^j}{\gamma_t(\tilde{x}_t^j)} \sum_i w_{t-1}^i p(\tilde{x}_t^j \mid x_{t-1}^i).$$

Here $(\tilde{w}_t^j, \tilde{x}_t^j)$ is the j^{th} weighted sample from the backward information filter at time t, i.e. approximating $\tilde{p}(x_t \mid y_{t:T})$. Thus, the smoother output is a re-weighted collection of the samples from the backward filter. This backward filter is approximated as

$$\begin{aligned} \tilde{p}(x_t \mid y_{t:T}) &\propto p(y_t \mid x_t) \tilde{p}(x_t \mid y_{t+1:T}) \\ &= p(y_t \mid x_t) \int \frac{p(x_t \mid x_{t+1}) \gamma_t(x_t)}{\gamma_{t+1}(x_{t+1})} \tilde{p}(x_{t+1} \mid y_{t+1:T}) dx_{t+1} \\ &\approx \gamma_t(x_t) p(y_t \mid x_t) \sum_i \tilde{w}_{t+1}^i \frac{p(x_t \mid x_{t+1}^i)}{\gamma_{t+1}(x_{t+1}^i)} \delta_{\{\tilde{x}_{t+1}^j\}}. \end{aligned}$$

Since this is very similar to the approximation used for the forward filter, many of the same techniques used there to improve performance can be applied. As with the forward-filter backward-smoother, the two-filter smoother takes $O(TN^2)$ time. [15] claim that the two-filter smoother is faster, has lower error and higher effective sample size than the forward-filter backward-smoother, but [145] does not find much to choose between them other than that the two filter smoother is somewhat faster to run.

In [145] a variation on the two-filter smoother is introduced that allows the positions of the smoother particles to be resampled from an arbitrary

proposal distribution, by using the following expansion in terms of *both* forward and backward filters in place of equation (2.33):

$$\begin{aligned} p(x_t \mid y_{1:T}) &\propto \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) dx_{t-1} \\ &\qquad \times p(y_t \mid x_t) \int \frac{p(x_t \mid x_{t+1}) \gamma(x_t)}{\gamma_{t+1}(x_{t+1})} \tilde{p}(x_{t+1} \mid y_{t+1:T}) dx_{t+1} \\ &\tilde{\propto} \sum_i w_{t-1}^i p(x_t \mid x_{t-1}^i) p(y_t \mid x_t) \sum_j \tilde{w}_{t+1}^j \frac{p(\tilde{x}_{t+1}^j \mid x_t)}{\gamma_{t+1}(\tilde{x}_{t+1}^j)}. \end{aligned}$$

Since this is given in terms of x_t it can be sampled using, for example, importance sampling or MCMC in order to generate new x_t samples from the smoother distribution. This will be an $O(N^2)$ operation at each time point, since evaluation of this distribution for any x_t is $O(N)$, but will generate a fresh set of samples to represent the smoothing distribution which might be better suited to the task if they can be chosen appropriately.

In [145], an importance sampling scheme similar to auxiliary particle filtering is used to choose N particle pairs (with one particle from each of the forward and backward filters) which can then be propagated to give samples of x_t with which to approximate the smoothing distribution. This formulation allows a neat approximation (approximating a sum over pair selection weights with its limiting integral) to find approximate marginal (rather than pairwise) weights for selecting each particle, allowing pairs to be drawn in $O(N)$ time, and giving this smoothing algorithm $O(TN)$ time complexity overall; see [145] for full details. An alternative fast smoothing scheme is given by [148], which can reduce the time cost of smoothing to $O(TN \log N)$ through the use of space partition trees.

2.3.4 Parameter Estimation

Parameter estimation using particle filters can be divided into two distinct flavours: batch estimation in which particle filters are used as a component in batch parameter estimation, and online estimation in which the parameters are estimated simultaneously with the state. A review of methods

of both types is given in [149].

Batch Methods

Particle filters can be used as components of pseudo-marginal MCMC schemes to perform exact inference, as seen in sections 2.1.5 and 2.2.3. These methods rest on the fact that, as seen in section 2.3.1, the particle filter can be used to give an unbiased estimate of observation likelihood. The main methods in this area are the Particle MCMC methods of [33] (especially the PMMH sampler and Particle Gibbs methods); these are examined in more detail in section 4.3.

An alternative to exact estimation of the parameters is to use particle filters as a means to do approximate parameter estimation. On first consideration such methods might seem of little interest when exact parameter estimation methods are available, however such approximate parameter estimates are often substantially quicker to calculate than exact estimates and so are of interest in situations where this is an important consideration. Some of these methods can be based on sufficient statistics calculated sequentially from the particle filter without storing the set of particle paths. In the case of huge datasets such methods might be the only computationally tractable options for parameter estimation. These methods are mainly concerned with finding point estimates of the parameters and work by using stochastic optimization techniques on the likelihood or posterior to find maximum likelihood or MAP parameter values. The main alternatives are stochastic gradient ascent and stochastic versions of the EM algorithm.

Stochastic EM algorithms were introduced by [150] and, in this context, replace the deterministic (and often intractable) E-step of the original algorithm with the calculation of a Monte Carlo approximation to the expectation $\mathbb{E}_{x|y,\theta^{(i)}}\left(\log p(x, y \mid \theta)\right)$ appearing in equation (2.18), i.e. using

$$\mathbb{E}_x\left(\log p(x, y \mid \theta) \mid y, \theta\right) \approx \sum_i w_i \log p(x^i, y \mid \theta), \tag{2.34}$$

where (w_i, x^i) are weighted Monte Carlo samples drawn from $p(x_{1:T} \mid y_{1:T}, \theta)$. While many methods can be used to generate the samples for the approximation (and even single samples can be used), particle filters lend themselves naturally to the use of their particle collection as such sample collections. In this case, the approximation in equation (2.34) is given by the weighted sum of logs of the full data likelihood for the particle filter which, for Markov models, is given by

$$\sum_i w_i \log p(x^i, y \mid \theta) = \sum_i w_i S_{i,T}(x^i_{1:T}, y_{1:T}, \theta)$$

where

$$S_{i,T}(x^i_{1:T}, y_{1:T}, \theta) = \sum_{t=1}^{T} \log p(y_t \mid x^i_t, \theta) + \sum_{t=2}^{T} \log p(x^i_t \mid x^i_{t-1}, \theta) + \log p(x^i_1 \mid \theta).$$

This can be written recursively as

$$S_{i,t}(x^i_{1:t}, y_{1:t}, \theta) = S_{i,t-1}(x^i_{1:t-1}, y_{1:t-1}, \theta) + \log p(y_t \mid x^i_t, \theta) + \log p(x^i_t \mid x^i_{t-1}, \theta),$$

which allows the expression in terms of θ to be built up while running the particle filter. If a sufficient statistic T_t is available for the parameters so that $p(\theta \mid T_t) = p(\theta \mid x_{1:t}, y_{1:t})$ and can be sequentially updated, then only this needs to be stored, rather than the entire path and observation history [151]. Only the final values of this statistic for each particle T^i_t will be necessary to form the required expectation. Finding such statistics is model dependent, but is often possible, e.g. [151] for bearings only tracking models and [152] for linear state models with non-linear observations.

The M-step of the algorithm then consists in maximizing this approximation with respect to θ. In some cases this will be straightforward to do analytically, but, as with EM, a generalized version of the algorithm can be produced by simply requiring the M-step to choose a new value of θ that increases this expression, allowing approximate or partial optimization methods to be used. The use of stochastic EM for batch parameter

estimation in non-linear state space models is examined in [153]; [154] uses it in an online setting.

The Monte Carlo approximation of the expectation in the E-step in equation (2.34) requires that the Monte Carlo samples approximate the distribution $p(x_{1:T} \mid y_{1:T}, \theta)$. However, as noted in section 2.3.3, the filter-smoother approximation of this becomes increasingly poor with increasing lag and thus becomes increasingly poor overall as the length of the time series increases. This leads the performance of the method to degrade when used with long time series [154]. A possible solution is to use a particle smoother to draw samples from $p(x_{1:T} \mid y_{1:T}, \theta)$ to create the sample collection; this approach is used in [145], making use of the $O(N)$ particle smoothing algorithm presented there, but could also be used with the backward-sampling method in [146] and variants thereof.

Gradient ascent approaches to parameter estimation are conceptually similar to stochastic EM, with an approximation of the gradient of the log-likelihood with respect to θ being calculated using the particle collection. This can be found using Fisher's identity [155]:

$$\nabla \log p(y \mid \theta) = \mathbb{E}_x \left(\nabla \log p(x, y \mid \theta) \mid y, \theta \right),$$

which makes the form of the approximation immediately clear by comparison to that derived for stochastic EM as

$$\nabla \log p(y \mid \theta) \approx \sum_i w_i \nabla S_{i,T}(x^i_{1:T}, y_{1:T}, \theta),$$

where $\nabla S_{i,T}$ is given by taking gradients of the individual terms in $S_{i,T}$. This expression shows the close link between gradient ascent and stochastic EM. If a gradient ascent step is used as the M-step in EM then they do the same thing. Examples of the gradient ascent approach applied to batch estimation are given in [156] and [157]. It is used for online estimation in [158], which gives extensive derivations of the log-likelihood and related terms. There it is noted that, while the standard particle filter estimate of

the likelihood $p(y_{1:T} \mid \theta)$ is unbiased, the estimate of the log likelihood is not, though this can be corrected as shown in [158].

Online Methods

The most obvious way to estimate parameters online in the particle filter framework is to simply add them to the state as static elements. However, this rapidly leads to degeneracy of the estimates, with values only being sampled for these parameters in the first period (since they remain constant from one period to the next). Subsequent resampling steps will lead to the loss of some of these values and the sample will steadily degenerate. Such early 'convergence' leads to the discarding of information arising from subsequent observations, and thus cannot be efficient.

This problem was recognized early in the development of particle filters and, in an attempt to overcome it, [61] and [159] sampled parameters from kernel densities as a way to reintroduce variety into the samples, effectively adding random jitter to the parameter samples. [57] gives a generalized version of these algorithms in an auxiliary particle filter, but acknowledge that the method "suffers from the obvious drawback that it throws away information about parameters in assuming them to be time-varying when they are, in fact, fixed".

An alternative scheme for introducing diversity into the parameter estimates is provided by the resample-move scheme [140] first encountered in section 2.3.2. This scheme can be used to rejuvenate the parameter samples in a similar way to the state rejuvenation shown in that section. In the parameter case, the target distribution for the MCMC rejuvenation is $p(\theta \mid x_{1:t}, y_{1:t})$. This parameter distribution may depend on the entire state and observation sequence so could take at least $O(t)$ time to calculate, which makes it unsuitable for use in a sequential method. However, it can be made suitable for sequential use if sequentially updateable sufficient statistics T_t are available such that $p(\theta \mid x_{1:t}, y_{1:t}) = p(\theta \mid T_t)$. This distribution can then be used as the target of the Markov chain using the acceptance

ratio

$$\alpha(\theta, \theta') = \frac{p(\theta' \mid T_t) q_\theta(\theta \mid \theta')}{p(\theta \mid T_t) q_\theta(\theta' \mid \theta)}$$

to sample new parameters θ'. This method is thus able to restore parameter sample diversity without introducing artificial dynamics. The method has proved popular, although as illustrated in [154] estimates produced in this way do not always converge to the correct values, and are inefficient because the particle filter is not a consistent estimator of the sufficient statistics $p(T_t \mid x_{1:t}, y_{1:t})$. This is because these are based on path estimates $p(x_{1:t} \mid y_{1:t})$ and, as noted in the previous section, these degenerate at long lags, leading to degraded estimates overall for long time series. This problem is much more serious for parameter and model estimation than for state estimation because these former depend on the entire state sequence whereas the state, if the dynamic model has good 'forgetting' properties (see e.g. [149]), only depends on relatively recent state values, which are well estimated.

A related approach to resample-move is to estimate both parameters and recent state using MCMC, similar to the approximate sequential MCMC type methods seen in section 2.3.2. Such methods include those of [151], [152] and [141] and, as seen above, are necessarily based on sufficient statistics in order to make them tractable.

The online approaches covered so far have tackled the problem of parameter estimation in a fully Bayesian way, but, as seen, these methods all suffer deficiencies. An alternative is to attempt to find point parameter estimates (either maximum likelihood or MAP) whilst using particle filtering to for state estimation. Here the gradient ascent and stochastic EM methods used for batch estimation can be adapted for sequential use. A simple approach is via a Robbins-Monro type stochastic approximation scheme [160], in which each iteration makes a move in the proposed direction weighted by a decreasing step size. This is the approach taken in [161] and [158],

which use gradient ascent with the update step

$$\theta_{t+1} = \theta_t + \eta_{t+1} \nabla \log p(y_t \mid y_{1:t-1})$$

where η_{t+1} is the (steadily decreasing) step size. The gradient of the marginal change to the log-likelihood can be calculated as

$$\nabla \log p(y_t \mid y_{1:t-1}) = \nabla \log p(y_{1:t}) - \nabla \log p(y_{1:t-1}),$$

although this estimate, along with the estimate of all additive functionals (i.e. functions $\sum_{k=1}^{t} \varphi(x_k)$ calculated sequentially, which includes many sufficient statistics) has variance that scales with $O(t^2)$ due to the particle degeneracy problem [162]. In [156; 162] an alternative method for calculating this is proposed whose variance only scales with $O(t)$, although this has quadratic complexity in the number of particles. A similar approach can be developed for EM algorithms; see [149] and the references therein. In fact a more general class of additive functionals can be calculated that effectively allows estimates to be based on an approximation of the forward-filter backward smoother estimates of state [163], albeit at $O(N^2)$ time cost (though in [163] it is suggested that this could be reduced to $O(N \log N)$ using space partition methods as in [148]).

[154] attempts to reduce the computational burden of a related method (applicable only in certain cases) through the introduction of a pseudo-likelihood function that shares a maximum with the true likelihood, but which can be calculated more efficiently through the use of a blocking idea on the past state and observations. Online EM is then used to optimize the resulting pseudo-likelihood estimate.

Online parameter estimation in particle filters remains an active area of research with no solution yet gaining complete acceptance or widespread use. The degeneracy problem produces what may be a fundamental limit for many methods, including the fully Bayesian approaches proposed to date. Some of the most recent maximum likelihood point estimation tech-

niques appear to offer a way around this problem, albeit at the price of computational complexity, and are therefore promising developments.

Chapter 3

Online Inference for Linear Jump Diffusion Models

This chapter concerns Bayesian filtering for continuous time linear jump diffusion models, for which efficient computational inference is possible. A state space approach is taken (see section 2.1), in which it is assumed that the dynamical model of the process under investigation follows a linear jump diffusion process. This chapter considers the problem of sequential filtering and backward sampling for these systems. The methods proposed are applied to various synthetic and real data, and a model is proposed for trend following in foreign exchange data. The problem of parameter estimation for such models is considered in chapter 4.

The class of models considered is restricted to those that, aside from their jump components, are linear Gaussian and time invariant, although this latter assumption could be relaxed within the framework presented. The time distribution of jumps can be freely chosen; jump sizes are assumed to be Gaussian, although this can be relaxed with a minor modification to the inference algorithm. The observation function is assumed to be a linear transform of the underlying process subject to additive Gaussian noise. These assumptions allow for efficient inference based on the Rao-Blackwellized particle filter.

The rest of the chapter is structured as follows. Section 3.1 briefly reviews some related work. Section 3.2 introduces the general dynamical model in the form of a governing multivariate stochastic differential equation. It shows how to solve this equation (in a distributional sense) in order to derive the state transition densities required by the filter. Section 3.3 presents the VRPF algorithm for the jump diffusion model. Section 3.4 shows how to draw samples from the smoother distribution. Section 3.5 presents details of a specific two factor model jump diffusion model for the purpose of trend following in finance. Section 3.6 gives the results of a number of tests of the algorithms presented throughout the chapter, along with the results of applying the trend following model to some foreign exchange data. Finally, section 3.7 draws conclusions from this work.

Some of the work in this chapter is based on my earlier work. The description of the model (section 3.2) and the variable rate particle filter (VRPF) algorithm (section 3.3) are based on those that appeared in [2] and [164]. The finance model (section 3.5) is based on the trend following model found in [2] and [164].

3.1 Related Work

Jump diffusion models have found their primary uses in finance, econometrics, physics and object tracking applications. In finance and econometrics they have been used to model the evolution of securities prices for option pricing, starting with [165], risk processes in insurance [166], credit risk [167] and the evolution of electricity spot prices [168; 169; 170]; in physics, they have been used to model neutron scattering e.g. [171]; and in object tracking the presence of jumps in a dynamic model has allowed the motion of maneuvering objects to be more accurately modelled [35; 172; 173]. Jump diffusion models can be useful in certain applications in place of regime switching models [168; 173]; in these cases jumps can be viewed as instantaneous switches into other dynamic regimes and, in the

case of multivariate state processes, can be used to reset or adjust some elements of the state process, allowing continuous variation of certain types of 'regime'. For example in the finance model presented in this chapter, jumps in the 'trend' process can be used to effectively switch the trending regime (in direction or in intensity); with a little modification, jumps can reset the trend process allowing them to switch from an existing trend to no prevailing trend.

The work in this chapter is closely related to tracking applications, since it is concerned with process inference from noisy observations via a state space approach. Financial applications, in which jump diffusion systems are most commonly encountered do not usually consider the possibility of noisy observations. [174] presents a particle filter based method for inference in general (non-linear) jump diffusion models. They note that the optimal filtering problem has "received little attention" for such systems.

The filtering method used here applies to linear jump diffusion systems and is based on the earlier work on *variable rate particle filters* (VRPFs) in [175; 35; 173; 172]. It makes use of Rao-Blackwellization to marginalize non-linear parts of the state, greatly improving efficiency and accuracy. By making the assumption that non-linear or non-Gaussian state transitions occur instantaneously at a countable number of random times, the variable rate particle filter can increase the number of elements in its stored state sequence only when one of these transitions occurs. The remaining state (between nonlinear transitions) can be inferred using the more accurate and computationally efficient Kalman filter. These ideas have been successfully applied to target tracking problems where the tracked objects are capable of executing sudden sharp maneuvers due to some internal or external thrust (modelled as the application of an instantaneous force), but follow simple linear dynamical models between thrusts [35; 173; 172].

The work here differs from some of this earlier work by introducing a slightly more general dynamical model, allowing jumps and diffusion components in all elements of the process. By assuming Gaussian jumps

it also allows more of the state to be marginalized out of the particle filter, requiring only jump times and the element of the state process they affect to be stored at each jump time. It also extends the methodology by presenting a straightforward and efficient method for drawing samples from the approximate smoother distribution; these are useful in some of the parameter estimation techniques presented in chapter 4. A specific instance of the model and inference algorithm in two state dimensions is applied to trend following in finance and this is demonstrated with an example of its application to foreign exchange data.

3.2 Model

The models considered are state space models of the form described in section 2.1, requiring a dynamical (state transition) model and an observation model in order to define them. This section outlines these for the jump diffusion processes under consideration and shows how the conditional transition model necessary for the Rao-Blackwellized filter can be calculated. As the systems being modelled are in continuous time, the most natural way to describe their state transitions is as stochastic differential equations (SDEs), the weak solution of which gives the state transition density.

3.2.1 Conditionally Linear State Transition Model

The linear time invariant jump diffusion state transition model for a K-dimensional process can be expressed as the multivariate SDE

$$dX_t = AX_t dt + B dW_t + C dJ_t, \tag{3.1}$$

where X_t is a K-dimensional system state vector, A, B and C are constant $K \times K$ matrices, dW_t is the instantaneous change of a Brownian motion diffusion process and dJ_t is the instantaneous change of a finite jump process defined below. The matrix A describes the interaction between the components of the state process, B is the Cholesky decomposition of the

covariance matrix of the diffusion process and C is the Cholesky decomposition of the jump size covariance matrix, assuming that jump sizes are Gaussian random variables. It is possible, within the inference framework presented here, to relax this final assumption to any jump size distribution, which can, if required, depend on the current time and system state.

In order to apply standard filtering techniques an expression for the distribution of the system state at a future time T, given its state (or state distribution) at the current time S ($S < T$) must be derived from this model. This can be done by first considering the diffusion system without the presence of jumps, which describes the system evolution *between* jumps, and then accounting for the jumps. Without jumps the system is governed by the SDE

$$dX_t = AX_t dt + B dW_t, \tag{3.2}$$

This system is linear time-invariant (LTI) Gaussian, meaning it can be solved in closed form using Itô calculus [176]; for LTI models this is a well known procedure. The solution (integrating from time S to T) is given by the stochastic integral

$$X_T = e^{A(T-S)}\left[X_S + \int_S^T e^{-At} B\, dW_t\right]. \tag{3.3}$$

If X_S is Gaussian distributed then X_T is itself Gaussian distributed because the stochastic integral is also a Gaussian random variable. Gaussian distributions are fully determined by their first two moments, so X_T is fully specified by its expectation and covariance. Since the expectation of the stochastic integral is zero the expectation of X_T is

$$\mathbb{E}(X_T) \;=\; e^{A(T-S)}\mathbb{E}(X_S). \tag{3.4}$$

The covariance of X_T can be found by noting the independence of X_S and $\int_S^T e^{-At} dW_t$ (since the integral depends on independent random innovations that occur after time S), and the fact that the latter integral has

an expectation of 0 in all components, so that some of the terms in the expansion given by

$$\mathrm{cov}(X_T) = \mathbb{E}\left((X_T - \mathbb{E}(X_T))\,(X_T - \mathbb{E}(X_T))'\right) \tag{3.5}$$

are equal to zero. Substituting in the expressions for X_T and $\mathbb{E}(X_T)$ in (3.3) and (3.4), respectively, and, for notational convenience, defining

$$Q(r,s) \;=\; \mathbb{E}\left(\left(\int_r^s e^{-At}B\,dW_t\right)\left(\int_r^s e^{-At}B\,dW_t\right)'\right), \tag{3.6}$$

this gives

$$\mathrm{cov}(X_T) = e^{A(T-S)}\Big[Q(S,T) + \mathrm{cov}(X_S)\Big](e^{A(T-S)})' \tag{3.7}$$

Using a multivariate instance of the Itô isometry applied to a deterministic process the expectation in the definition of Q in equation (3.6) can be calculated as

$$Q(r,s) = \int_r^s e^{-At}BB'(e^{-At})'dt, \tag{3.8}$$

giving a deterministic expression for the covariance of X_T. The calculation of the integral expression for $Q(r,s)$ in (3.8) is not completely trivial but can be obtained using matrix fraction decomposition [177] or by series expansion of the exponential functions [172; 164]. The series expansion is only plausible for low dimensional systems and relies on the numerically unstable calculation of the Jordan normal form of the A matrix in the case when A is not diagonalizable, so the former method is preferred for its generality and superior numerical stability. Appendix B gives further details on calculation of this integral.

Without jumps, then, the transition density is given by

$$p(X_T \mid X_S) \sim \mathcal{N}\big(\mathbb{E}(X_T|X_S), \mathrm{cov}(X_T|X_S)\big) \tag{3.9}$$

where $\mathcal{N}(\mu,\Sigma)$ indicates a multivariate Gaussian distribution of mean μ

and covariance Σ. The conditional expectation and covariance in equation (3.9) are given by the expressions in equations (3.4) and (3.7).

No closed form solution for the system with jumps in equation (3.1) system exists in general. However, by conditioning on jump times the system can be treated as if jump times are known *a priori*. This allows separation of the system into a tractable LTI Gaussian part (time between jumps), solved as shown above, and a nonlinear, non-Gaussian part (the jumps).

The arrival process for jumps can be chosen freely. Care must be taken to distinguish which state component(s) a given jump occurred in. In what follows, this is achieved through the indicator functions $\mathcal{I}_{\text{jump}_i(\tau_k)}$ which indicate whether the k^{th} jump occurred in the i^{th} state component by taking the value 1 when the jump is in the specified component and 0 otherwise. The time of the jump of the k^{th} jump is denoted τ_k.

Jump sizes S_k are modelled as following a multivariate Gaussian distribution so that

$$S_k \sim \mathcal{N}(0, D_{j_k}), \tag{3.10}$$

with

$$D_{j_k} = \text{diag}\left(\mathcal{I}_{\text{jump}_1(\tau_k)}, ..., \mathcal{I}_{\text{jump}_K(\tau_k)}\right)$$

This allows the jump process J_t to be defined as

$$J_t = \sum_{k \in \{k | \tau_k < t\}} S_k \tag{3.11}$$

so that $dJ_t = S_k$ at τ_k.

A straightforward choice of jump arrival process is to have the jumps in each component follow an independent Poisson arrival process with rate

λ_i. This gives

$$\begin{aligned}\tau_k - \tau_{k-1} &\sim \min\left(\text{exponential}\left(\lambda_1\right), \text{exponential}\left(\lambda_2\right)\right) \\ &= \text{exponential}\left(\lambda_1 + \lambda_2\right).\end{aligned}$$

There is no necessity to limit the jump process to an exponential inter-arrival distribution; for example, gamma distributions were used in some earlier tracking work [172]. Indeed, non-memoryless jump distributions could be used to create processes with time-varying stochastic volatility by, for example, causing jumps to cluster in areas of high activity.

Calculation of the state transition density conditional on knowing the jump types and times can be accomplished by treating the transition as multiple segments of different types, defined by the jumps. For example, if a single jump occurs between times S and T at time $\tau \in (S, T]$, the transition can be though of as occurring in three parts: a pre-jump diffusion from S to τ, the jump itself and a post-jump diffusion from τ to T. With no jumps or multiple jumps between S and T this can be modified appropriately by considering a single diffusion section from S to T or multiple diffusion and jump sections, respectively.

Following the diffusion from S to τ (where τ is the instant before the jump occurs) the conditional distribution of the state can be calculated exactly as in the non-jumping case above, giving a Gaussian state distribution with first and second moments given by

$$\begin{aligned}\mathbb{E}(X_\tau|\tau, X_S) &= e^{A(\tau-S)}X_S \\ \text{cov}(X_\tau|\tau, X_S) &= e^{A(\tau-S)}Q(S,\tau)(e^{A(\tau-S)})',\end{aligned}$$

using the formulae in equations (3.4) and (3.7), respectively.

Across the period of the jump, from time τ to τ^+ (where τ^+ is the instant immediately after the jump occurs) the state distribution can be calculated by noting that jump sizes are assumed to be Gaussian distributed with zero mean and independent of any other innovations as in (3.37), giving a Gaus-

sian post-jump distribution given by

$$\begin{aligned}\mathbb{E}(X_{\tau^+}|\tau, X_S) &= \mathbb{E}(X_\tau|\tau, X_S)\\ \operatorname{cov}(X_{\tau^+}|\tau, X_S) &= \operatorname{cov}(X_\tau|\tau, X_S) + \Sigma_{J_\tau}.\end{aligned}$$

where the jump covariance matrix Σ_{J_τ} is given by

$$\Sigma_{J_\tau} = D_{j_\tau} C C' D_{j_\tau}.$$

The expressions in equations (3.4) and (3.7) can then be used to calculate the first two moments of the Gaussian state distribution following the diffusion from τ^+ to T, which are given by

$$\begin{aligned}\mathbb{E}(X_T) &= e^{A(T-\tau)}\mathbb{E}(X_{\tau^+}|\tau, X_S),\\ \operatorname{cov}(X_T) &= e^{A(T-\tau)}\Big[Q(\tau, T) + \operatorname{cov}(X_{\tau^+}|\tau, X_S)\Big](e^{A(T-\tau)})'.\end{aligned}$$

Written in terms of X_S only, these moments are given by

$$\mathbb{E}(X_T|\tau, X_S) = e^{A(T-S)}X_S, \tag{3.12}$$

which does not depend on the jump, so is the same for any number of jumps in the period, and

$$\begin{aligned}\operatorname{cov}(X_T \mid \tau, X_S) &= e^{A(T-\tau)}\Big[Q(\tau, T) + \Sigma_{J_\tau}\Big](e^{A(T-\tau)})'\\ &\quad + e^{A(T-S)}Q(S, \tau)(e^{A(T-S)})'.\end{aligned} \tag{3.13}$$

For an arbitrary set of jumps between S and T, this is given by

$$\begin{aligned}\operatorname{cov}(X_T \mid \mathcal{T}_{S:T}, X_S) &= \sum_{i=2}^{|\mathcal{T}|-1} e^{A(T-\mathcal{T}_i)}[Q(\mathcal{T}_i, \mathcal{T}_{i+1}) + \Sigma_{J_i}](e^{A(T-\mathcal{T}_i)})'\\ &\quad + e^{A(T-S)}Q(S, \mathcal{T}_1)(e^{A(T-S)})',\end{aligned} \tag{3.14}$$

where $\mathcal{T}_{S:T}$ is an ordered list of the jump times from S to T, augmented with

the final time T, so that $\mathcal{T}_i$ is the i^{th} jump between S and T and, if there are n jumps in total, $\mathcal{T}_{n+1} = T$; Σ_{J_i} is the covariance matrix corresponding to the i^{th} jump.

These allow the required conditional state transition density (conditional the jump times) to be defined as

$$p(X_T \mid \mathcal{T}_{S:T}, X_S) \sim \mathcal{N}\left(\mathbb{E}(X_T \mid \mathcal{T}_{S:T}, X_S), \operatorname{cov}(X_T \mid \mathcal{T}_{S:T}, X_S)\right) \tag{3.15}$$

3.2.2 Observation Model

In the models considered here a linear observation model is used, so that an observation y_i made at time t_i can be expressed as a linear function of the system state corrupted by additive Gaussian noise, i.e.

$$y_i = HX_{t_i} + \epsilon_i,$$

with $\epsilon_t \sim \mathcal{N}(0, \Sigma_{\text{obs}})$. As per the standard state space model assumption shown in figure 2.1, observations y_i are assumed to be conditionally independent of all other observations and elements of the state process given the value of the state process at the time of the observation X_t.

For such linear Gaussian observation models, Rao-Blackwellization, as covered in section 2.3.2, allows the entire state other than the jump times to be inferred using the Kalman filter, since conditional on the jump times, the system is linear Gaussian. For observation models that are themselves nonlinear or non-Gaussian in some components, more components of the state will need to be inferred using the particle filter.

If nonlinear or non-Gaussian observation functions are necessary, the inference algorithm presented below in section 3.3 requires some modification, with more components of the state needing to be estimated via the particle filter. In such cases, though, the analytical methods for weight update calculation in e.g. [129] (see section 2.3.2) will be applicable between jumps.

3.3 State Inference

The inference algorithm must take a series of observations $y_{1:N} = \{y_i \mid i \in 1, ..., N\}$ and use these to infer the posterior filtering distribution of the underlying state X_t at time t, along with the set of jump times τ and their types. Rao-Blackwellization means that only jumps need be estimated by the particle filter. Due to the random arrival of jumps, different state paths will include different numbers of jumps meaning that the non-linear state space can have varying dimension between particles. This section shows how the variable rate particle filter of [35] can be applied to this problem, allowing particles to consist of state spaces (jump collections) of varying sizes.

Given inferred jump times, the filtering distribution of the process state $p(X_{t_j} \mid y_{1:j})$ at time t_j can be inferred using the Kalman filter to infer the filtering distribution for X_{t_j} conditioned on the set of jumps of each particle in the filter. The posterior filtering distribution for the state process is then given by the weighted sum of these distributions and is thus approximated by the Gaussian mixture

$$p(X_{t_j} \mid y_{1:j}) \approx \sum_{p \in P_{t_j}} w_{t_j}^p p(X_{t_j}^p \mid y_{1:j}, \mathcal{T}_{t_0:t_j}^p), \tag{3.16}$$

$$\approx \sum_{p \in P_{t_j}} w_{t_j}^p \mathcal{N}(\mu_{j|0:j}^p, C_{j|0:j}^p). \tag{3.17}$$

where $p(X_{t_j}^p \mid y_{1:t_j}, \mathcal{T}_{t_0:t_j}^p)$ is the posterior filtering density obtained from the Kalman filter conditioned on particle p's set of jump times between t_0 and t_j; this has mean $\mu_{j|0:j}^p$ and covariance $C_{j|0:j}^p$ given by equations (2.6) and (2.7). It is useful to store these filter mean and covariances for each particle to allow them to be sequentially updated.

3.3.1 Variable Rate Particle Filter (VRPF)

In order to infer the marginalized jump times and types, the variable rate particle filter algorithm of [35] is used. In what follows, it is assumed

that a collection of particles is available representing the posterior filtering density at t_{j-1}, the time of the $(j-1)^{\text{th}}$ observation (in the case where the state prior is specified at the initial observation time $t_0 = t_1$, this can be a single particle with an empty set of jumps and the state mean and covariance equal to their prior values). When a new observation is received at t_j this particle collection must be updated to a new one that represents the posterior filtering density after making that observation. Figure 3.1 shows this process in outline for a single particle in the collection and algorithm 3 briefly outlines the algorithm.

The first step in updating each particle is to decide the number of successor particles (children) M_j^p each particle p will have in generation j (step 1 in figure 3.1 and algorithm 3). This is performed by a residual resampling step (see section 2.3.1), in which

$$M_j^p = \lfloor N w_{t_{j-1}}^p \rfloor + S^{(p)} \tag{3.18}$$

with $S^{(p)} \sim \mathcal{M}(N-R, \bar{w}_{j-1}^{1:N_{j-1}})$, where $R = \sum_{p=1}^{N} \lfloor N w_{t_{j-1}}^p \rfloor$, $\bar{w}_{j-1}^i = \frac{N w_{t_{j-1}}^p - \lfloor N w_{t_{j-1}}^p \rfloor}{N-R}$ for all $p = 1, ..., N$ and $\mathcal{M}(\cdot, p_{1:M})$ is the multinomial distribution with selection probabilities $p_1, p_2, ...p_M$. N here is the target number of particles in each generation, and N_{j-1} is the number of actual particles in generation $j-1$, which might be different from the target N because identical non-jumping particles can be collapsed (see below) in order to save computation.

Initially, each child particle is a copy of its parent, with the same jumps from time t_0 to t_{j-1}, and the parent's weight is divided evenly, so that the weight of each offspring particle q is

$$w_{t_{j-1}}^q = \frac{w_{t_{j-1}}^p}{M_j^p}. \tag{3.19}$$

Once the number offspring has been chosen, new jumps (times and types) τ_*^q are sampled for each child q (step 2 in figure 3.1 and algorithm 3)

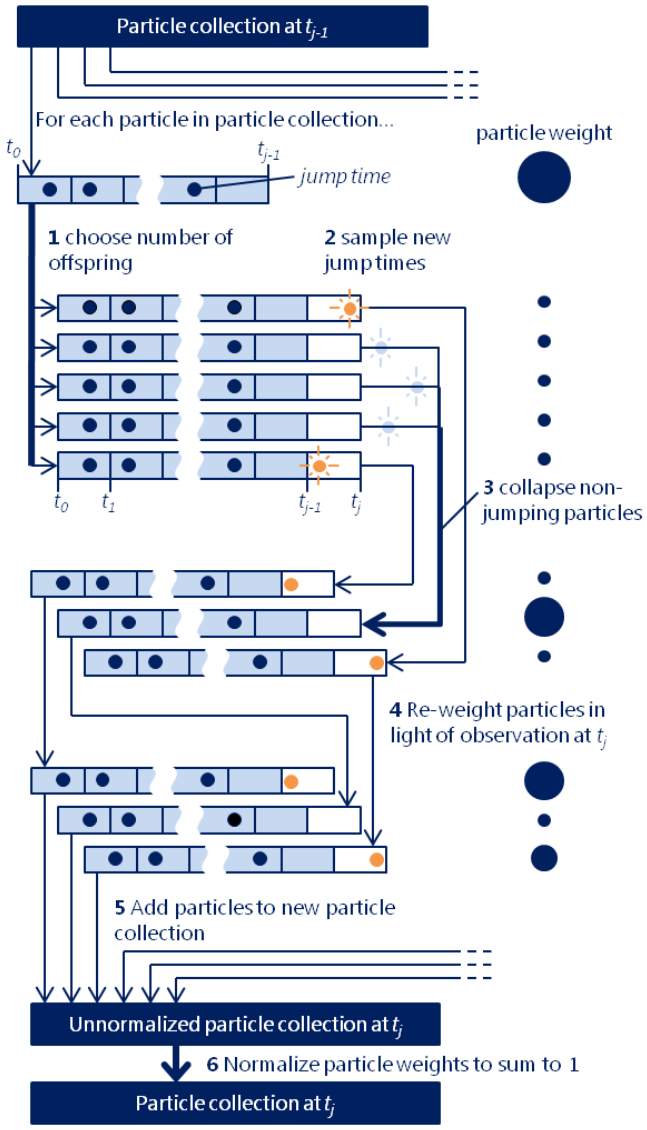


Figure 3.1: An update step for a single particle of the variable rate particle filter. Shaded rectangles represent particles consisting of collections of jumps represented by dots. Representative particle weights are illustrated along the right-hand side by circles, with diameter corresponding to weight. See text for further information on numbered steps

from a jump proposal distribution, given the parent's jump times and types $\mathcal{T}^{p}_{t_0:t_{j-1}}$, so that

$$\tau^{q}_{*} \sim q_{\text{jump}}(\tau_* \mid t_{j-1}, \mathcal{T}^{p}_{t_0:t_{j-1}}). \tag{3.20}$$

The use of jump distributions other than that of the model is particularly useful for processes with very rare jumps, since near-term jumps can be proposed and only accepted if deemed likely after incorporating the information from the latest observation.

Some of these newly proposed jump times will occur before the current observation time t_j (e.g. the orange jumps in the first and last particle in step 2 of figure 3.1); such offspring are termed *jumping* particles, since they contain a jump between the previous and current observations. The other particles have proposed jumps beyond the current observation time and are called *non-jumping*. For jumping particles, jumps continue to be sampled until no more are proposed before the current observation time, allowing for the possibility of multiple jumps between observations.

The next step (step 3 in figure 3.1 and algorithm 3) collapses all non-jumping offspring into a single offspring particle. This is possible since all non-jumping particles are identical up to the time of the current observation t_j, with a set of jump times the same as that of their parent particle. This step is different from most resampling schemes because in most applications there are not large numbers of identical offspring. Here, however, this fact allows calculation to be streamlined by only having one particle representing all identical offspring. The weight of the single particle into which all non-jumping particles is collapsed is the sum of the weight of all non-jumping particles, so that

$$w^{\text{NJ}(p)}_{t_{j-1}} = \frac{N_0}{M^{p}_{j}} w^{p}_{t_{j-1}},$$

where $\text{NJ}(p)$ is the single representative non-jumping offspring of parent p and N_0 is the number of non-jumping offspring originally sampled (of M^{p}_{j} offspring).

The particles of this reduced particle set (containing a number of jumping particles and at most one non-jumping particle) are then re-weighted in light of the observation at t_j (step 4 in figure 3.1 and algorithm 3). For the Rao-Blackwellized filter used here, the weight update is given by

$$w^q_{t_j} \propto w^q_{t_{j-1}} \frac{p(\mathcal{T}^q_{t_{j-1}:t_j}|t_{j-1}, \mathcal{T}^q_{t_0:t_{j-1}})p(y_j|y_{1:j-1}, \mathcal{T}^q_{t_0:t_j})}{q_{\text{jump}}(\mathcal{T}^q_{t_{j-1}:t_j}|t_{j-1}, \mathcal{T}^q_{t_0:t_{j-1}})}, \tag{3.21}$$

where $\mathcal{T}^q_{s:t}$ is the set of jumps of particle q between times s and t, and $p(\mathcal{T}^q_{t_{j-1}:t_j}|t_{j-1}, \mathcal{T}^q_{t_0:t_{j-1}})$ is the model's jump distribution, given the set of jumps $\mathcal{T}^q_{t_0:t_{j-1}}$ at time t_{j-1} . The denominator here is the proposal density for the jumps used in equation (3.20). Using a jump proposal function equal to the conditional jump distribution gives the bootstrap filter with the correspondingly simpler weight update

$$w^q_{t_j} \propto w^q_{t_{j-1}} p(y_j|y_{1:j-1}, \mathcal{T}^q_{t_0:t_j}). \tag{3.22}$$

The term $p(y_j|y_{1:j-1}, \mathcal{T}^q_{t_0:t_j})$ is the observation likelihood given the jump time collection and can be obtained from the PED of the Kalman filter in equation (2.9). Once this has been done for all offspring particles, they can be added to the new particle collection for time t_j (step 5 in figure 3.1 and algorithm 3).

This process is repeated for all particles in the time t_{j-1}posterior particle collection, giving a new particle collection representing the posterior filtering distribution at time t_j. The weights of the particles are given by normalizing them so that they sum to 1 (step 6 in figure 3.1 and algorithm 3).

3.4 Backward Sampling for VRPF

For some versions of the Particle MCMC parameter estimation algorithms described in chapter 4 it is useful to be able to draw a sample from the (approximate) smoothing distribution $\hat{p}(X_{t_{1:N}}, \mathcal{T}_{t_0:t_N} \mid y_{1:N})$. Such samples can be obtained by using the forward-filtering backward-sampling approach of

Algorithm 3 Variable rate particle filter (VRPF) algorithm outline

Initialization ($j = 0$): Create initial particle collection P_{t_0} of 1 particle with no jumps
while *observations available* **do**
  $j = j + 1$
  Observe y_j
  0 Initialize a new (empty) particle collection, $P_{t_j} = \emptyset$
  foreach *particle* $p \in P_{t_{j-1}}$ *(the previous particle collection)* **do**
    1a Choose the number of successor particles M_j^p for p using a residual resampling scheme; see equation (3.18)
  end
  foreach *particle* $p \in P_{t_{j-1}}$ *(the previous particle collection)* **do**
    1b Assign an equal share of particle p's weight to all children
    Initialize a set of particle p's children, $Q_p = \emptyset$
    Initialize count of non-jumping children $N_0 = 0$
    foreach *successor particle* q **do**
      2a Sample new jumps τ_*^q for the particle from the jump proposal distribution in equation (3.20)
      if $\min(\tau_*^q) < t_j$ **then**
        2b add the jump to the particle's set of jumps
        2c add the particle to children i.e. $Q_p := Q_p \cup q$
        2d sample any further jumps before t_j
      else
        2d note the particle as non-jumping; $N_0 := N_0 + 1$
      end
    end
    3a Collapse non-jumping successor particles into a single particle. i.e. create a particle q_0 with the same jumps as its parent p and weight equal to the share of the parent's weight due to non-jumping children.
    (set $\tau_{t_0:t_j}^{q_0} = \tau_{t_0:t_{j-1}}^{p}$ and $w_{t_{j-1}}^{q_0} = w_{t_{j-1}}^{p} N_0 / M_j^p$)
    3b add this particle q_0 to the set of children, $Q_p = Q_p \cup q_0$
    foreach *child particle* $q \in Q_p$ *(set of children)* **do**
      4 Re-weight particle q in light of current observation as in equation (3.22)
    end
    5 Add all child particles to new particle collection $P_{t_j} = P_{t_j} \cup Q_p$
  end
  6 Normalize particle weights in new particle collection so they sum to 1
  Result: New particle collection represents posterior filtering density after seeing given observation; this becomes current particle collection
end

[146], as described in section 2.3.3.

For notational clarity, it will be assumed in this section that observations occur at integer times, so that $t_i = i$, though this assumption can easily be relaxed. Furthermore, let $\mathcal{T}$ be the entire set of jumps occurring from t_0 to t_N.

3.4.1 Using Final Filter Jumps

The simplest approach to obtaining a sample from the joint smoother distribution for the jumps (times and types) $\mathcal{T}$ and the process state at all observation times $X_{1:T}$ is to use the output of the final stage of the particle filter as an approximate smoother sample of the jumps, i.e. to use the filter-smoother for the jump times. A sample can be drawn from the approximate smoother distribution

$$\hat{p}(\mathcal{T} \mid y_{1:N}) = \sum_i w_T^i \delta_{\{\mathcal{T}^i\}}$$

where $\mathcal{T}^i$ is the set of all jumps of particle i, by selecting a particle from the final particle collection, according to the particle weights w_T^i and taking that particle's entire jump history as the sampled value $\mathcal{T}^* = \mathcal{T}^i$. This can be used to obtain a joint sample of jumps and process states using the decomposition

$$\hat{p}(X_{1:N}, \mathcal{T} \mid y_{1:N}) = \hat{p}(\mathcal{T} \mid y_{1:N}) p(X_{1:N} \mid \mathcal{T}, y_{1:N}),$$

and the fact that given a set of jump times $\mathcal{T}^*$ it is possible to sample the process states exactly from their smoother distribution $p(X_{t_{1:N}} \mid y_{1:N}, \mathcal{T}^*)$. This is done using a standard backward sampling result adapted to account for jumps, conditioned on the sampled jump times collection, and is described in the rest of this section. The superscript $*$ is used to indicate already-sampled values of the corresponding variables.

The standard approach to backward sampling for linear Gaussian mod-

els [178; 179; 180] can be straightforwardly adapted to account for the presence of jumps, based on the decomposition

$$p(X_{1:N} \mid y_{1:N}, \mathcal{T}) = p(X_N \mid y_{1:N}, \mathcal{T}) \prod_{n=1}^{N-1} p(X_n \mid X_{n+1}, y_{1:n}, \mathcal{T}_{1:n+1}).$$

This relies on the conditional independence structure of the model for the fact that $p(X_n \mid X_{n+1:N}, y_{1:N}, \mathcal{T}) = p(X_n \mid X_{n+1}, y_{1:n}, \mathcal{T}_{1:n+1})$. Given a sample of the subsequent state X^*_{n+1}, a sample of X^*_n can be drawn from the distribution $p(X_n \mid X^*_{n+1}, y_{1:n}, \mathcal{T}^*_{1:n+1})$, which is given by

$$p(X_n \mid X^*_{n+1}, y_{1:n}, \mathcal{T}^*_{1:n+1}) \propto p(X^*_{n+1} \mid X_n, \mathcal{T}^*_{n:n+1}) p(X_n \mid y_{1:n}, \mathcal{T}^*_{1:n+1}),$$

where $p(X_n \mid y_{1:n}, \mathcal{T}^*_{1:n+1})$ is the filtering distribution for X_n conditional on the sampled jump times $\mathcal{T}^*_{1:n+1}$. For linear Gaussian models, this is Gaussian and can be obtained using the Kalman filter. Thus $p(X_n \mid y_{1:n}, \mathcal{T}^*_{1:n+1}) = \mathcal{N}\left(X_n; \mu^*_{n|n}, \Sigma^*_{n|n}\right)$, where $\mu^*_{n|n}$ and $\Sigma^*_{n|n}$ are the mean and covariance calculated from the Kalman filter after the n^{th} observation using the jump set $\mathcal{T}^*$. The transition density $p(X^*_{n+1} \mid X_n, \mathcal{T}^*_{n:n+1})$ is that from equation (3.15), again a Gaussian, $\mathcal{N}\left(X^*_{n+1}; F^*_n X_n, Q^*_n\right)$, where F^*_n and Q^*_n are the state transition and covariance matrices from n to $n+1$ corresponding to jumps $\mathcal{T}^*$ and defined by equations (3.12) and (3.14).

Algebraic manipulation using the Gaussian distribution identities in appendix A allows the backward sampling distribution of X_n to be found in terms of these known quantities.

$$\begin{aligned} p(X_n \mid X^*_{n+1}, y_{1:n}, \mathcal{T}^*_{1:n+1}) &\propto \mathcal{N}\left(X^*_{n+1}; F^*_n X_n, Q^*_n\right) \mathcal{N}\left(X_n; \mu^*_{n|n}, \Sigma^*_{n|n}\right) \\ &\propto \mathcal{N}(X_n \mid \mu, \Sigma) \end{aligned} \tag{3.23}$$

with

$$\begin{aligned} \Sigma^{-1} &= (F^*_n)'(Q^*_n)^{-1} F^*_n + (\Sigma^*_{n|n})^{-1}, \\ \mu &= \Sigma\left((F^*_n)'(Q^*_n)^{-1} X^*_{n+1} + (\Sigma^*_{n|n})^{-1} \mu^*_{n|n}\right). \end{aligned}$$

This distribution gives the conditional smoother distribution for X_n given a sample X^*_{n+1}, so can be used to sample X_n by drawing from the Gaussian distribution in equation (3.23). The backward sampling process is started with a sample of X_N drawn from the final Kalman filter distribution $\mathcal{N}\left(X_N; \mu_{N|N}, \Sigma_{N|N}\right)$.

3.4.2 Backward Sampling of Jumps

In order to improve the diversity of samples generated by backward sampling, jumps and process states can be jointly back-sampled using the particle filter output and an algorithm similar to the forward-filtering backward-sampling algorithm of [181; 146] (see section 2.3.3).

The joint smoother density of the jumps and process states can be written as

$$p(X_{1:N}, \mathcal{T} \mid y_{1:N}) = p(X_N, \tau_N \mid y_{1:N}) \prod_{n=1}^{N-1} p(X_n, \tau_n \mid X_{n+1}, \tau_{n+1:N}, y_{1:n}),$$

where $\tau_n = \mathcal{T}_{n-1:n}$ is the collection of jumps from $n-1$ to n (and hence τ_1 does not exist). This relies on the model structure for the fact that $p(X_n, \tau_n \mid X_{n+1:N}, \mathcal{T}_{n:N}, y_{1:N}) = p(X_n, \tau_n \mid X_{n+1}, \mathcal{T}_{n:N}, y_{1:n})$, although note that the structure of the jump process has not been assumed to be Markovian.

The distribution $p(X_N, \tau_N \mid y_{1:N})$ is approximated by the final filter density from the particle filter as

$$p(X_N, \tau_N \mid y_{1:N}) \approx \sum_i w_N^i \delta_{\{\tau_N^i\}} \mathcal{N}\left(X_N; \mu_{N|N}^i, \Sigma_{N|N}^i\right).$$

This distribution can be sampled as $\hat{p}(\tau_N \mid y_{1:N})\hat{p}(X_N \mid \tau_N, y_{1:N})$ by first sampling a particular $\tau_N^* = \tau_N^i$ from the particle approximation $\hat{p}(\tau_N \mid y_{1:N})$ by choosing a particle according to the final particle weights w_T^i. X_N can then be sampled from a Gaussian distribution with mean and covariance corresponding to the selected particle, sampling X_N^* from $\hat{p}(X_N \mid \tau_N^*, y_{1:N}) = \mathcal{N}\left(X_N; \mu_{n|n}^i, \Sigma_{n|n}^i\right)$.

Subsequent samples are then drawn from the following distribution, with $\mathcal{T}^*_{N-1:N} = \tau^*_N$.

$$\begin{aligned} p(X_n, \tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}) &\propto p(X^*_{n+1}, \mathcal{T}^*_{n:N} \mid X_n, \tau_n, y_{1:n}) p(X_n, \tau_n \mid y_{1:n}) \\ &= p(X^*_{n+1} \mid \tau_{n+1}, X_n) p(\mathcal{T}^*_{n:N} \mid \tau_n) p(X_n, \tau_n \mid y_{1:n}). \end{aligned}$$

Once again, $p(X_n, \tau_n \mid y_{1:n})$ is approximated by the particle collection after observation n, so that

$$\begin{aligned} \hat{p}(X_n, \tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}) \propto \sum_i & p(X^*_{n+1} \mid \tau_{n+1}, X_n) p(\mathcal{T}^*_{n:N} \mid \tau_n) \\ & \times w^i_n \delta_{\{\tau^i_n\}} \mathcal{N}\left(X_n; \mu^i_{n|n}, \Sigma^i_{n|n}\right). \end{aligned}$$

The transition density $p(X^*_{n+1} \mid \tau_{n+1}, X_n)$ is that from equation (3.15), given by the Gaussian $\mathcal{N}\left(X^*_{n+1}; F_n X_n, Q_n\right)$, with F_n and Q_n defined as in the preceding section. This allows the above distribution (denoted by $\hat{p}_n$ for brevity) to be written, via the identity in equation (A.2), as

$$\begin{aligned} \hat{p}_n &\propto \sum_i \mathcal{N}\left(X^*_{n+1}; F_n X_n, Q_n\right) p(\mathcal{T}^*_{n:N} \mid \tau^i_n) w^i_n \delta_{\{\tau^i_n\}} \mathcal{N}\left(X_n; \mu^i_{n|n}, \Sigma^i_{n|n}\right) \\ &= \sum_i W_i \, \mathcal{N}\left(X_n; \mu_i, \Sigma_i\right) \delta_{\{\tau^i_n\}} \end{aligned} \tag{3.24}$$

where

$$\begin{aligned} W_i = & \frac{w^i_n |\Sigma_i|^{\frac{1}{2}} |F_n| p(\mathcal{T}^*_{n:N} \mid \tau^i_n)}{(2\pi)^{\frac{k}{2}} |\Sigma^i_{n|n}|^{\frac{1}{2}} |Q_n|^{\frac{1}{2}}} \\ & \times \exp\left[-\frac{1}{2}\left((X^*_{n+1})' Q_n^{-1} X^*_{n+1} + \mu'_{n|n} \Sigma^{-1}_{n|n} \mu_{n|n} - \mu'_i \Sigma_i^{-1} \mu_i\right)\right] \end{aligned} \tag{3.25}$$

and

$$\Sigma_i^{-1} = F_n' Q_n^{-1} F_n + (\Sigma^i_{n|n})^{-1} \tag{3.26}$$

$$\mu_i = \Sigma_i \left(F_n Q_n^{-1} X^*_{n+1} + (\Sigma^i_{n|n})^{-1} \mu^i_{n|n}\right). \tag{3.27}$$

Using the decomposition

$$\hat{p}(X_n, \tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}) = p(X_n \mid X^*_{n+1}, \tau_n, y_{1:n})\hat{p}(\tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}),$$

a joint sample of τ_n and X_n can be drawn by first drawing τ^*_n from the approximate smoother distribution $\hat{p}(\tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n})$, followed by drawing X^*_n from $p(X_n \mid X^*_{n+1}, \tau^*_n, y_{1:n})$. An expression for $\hat{p}(\tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n})$ can be found by integrating the expression in equation (3.24) over X_n, so that

$$\begin{aligned} \hat{p}(\tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}) &= \int \hat{p}(X_n, \tau_n \mid X^*_{n+1}, \mathcal{T}^*_{n:N}, y_{1:n}) dX_n \quad (3.28) \\ &= \frac{1}{\sum_i W_i} \sum_i W_i \, \delta_{\{\tau^i_n\}}. \quad (3.29) \end{aligned}$$

This distribution is easily sampled by calculating W_i for each particle i in the particle collection, normalizing these so they sum to 1 and then selecting a particle according to the weights thus calculated. The sample τ^*_n is taken to be that particle's jumps, i.e. $\tau^*_n = \tau^j_n$ if particle j is selected. Given the sample τ^*_n, a sample X^*_n is drawn from $p(X_n \mid X^*_{n+1}, \tau^*_n, y_{1:n})$ as given in equation (3.23), with $\mu_{n|n}$ and $\Sigma_{n|n}$ corresponding to the selected particle. The overall jump sample is updated as $\mathcal{T}^*_{n-1:N} := \tau^*_n \cup \mathcal{T}^*_{n:N}$.

This sampling procedure is repeated until a sample X^*_1 is drawn, at which point a complete sample of the jumps and state sequence will have been drawn from the smoothing distribution. The full process is outlined in algorithm 4.

For jumps distributed exponentially in time, the distribution $p(\mathcal{T}^*_{n+1:N} \mid \tau^i_n)$ is the same for each particle, owing to the memoryless property of the exponential distribution and so does not need to be calculated since it can be considered part of the proportionality constant and will be corrected for by the normalization in equation (3.29).

The algorithm in this section is essentially the same as the JBS-RBPS algorithm in the technical report [182] (using Algorithm 2 in that report for

Algorithm 4 Backward sampling of jump times from approximate smoother distribution $\hat{p}(X_{1:N}, \mathcal{T} \mid y_{1:N})$

1 Run VRPF algorithm (algorithm 3) to obtain particle collections $P_{1:N}$ representing the approximate posterior filtering densities $\hat{p}(X_n, \tau_n \mid y_{1:n})$ after each observation n

2 Sample final period values:

- **2a** Select a particle j from the final particle collection P_N from the particle filter with probability according to its weight
- **2b** Set $\tau^*_N = \tau^j_N$, i.e. jumps of particle j from $N-1$ to N; $\mathcal{T}^*_{N-1:N} = \tau^*_N$
- **2c** Sample $X^*_N \sim \mathcal{N}\left(\mu^j_{N|N}, \Sigma^j_{N|N}\right)$

while $n \geq 1$ **do**

3 Evaluate W_i (equation (3.25)) for each particle i in the particle collection P_n

4 Sample period n values:

- **4a** Normalize the W_i so they sum to 1; see equation (3.29)
- **4b** Using normalized W_i as weights, select a particle j
- **4c** Set $\tau^*_n = \tau^j_n$, i.e. jumps of particle j from $n-1$ to n; $\mathcal{T}^*_{n:N} = \mathcal{T}^*_{n-1:N} \cup \tau^*_n$
- **4d** Sample $X^*_n \sim \mathcal{N}\left(\mu^j_{n|n}, \Sigma^j_{n|n}\right)$; see equations (3.27) and (3.26)

5 $n := n - 1$

end

OUTPUT: $\mathcal{T}^*_{1:N}$ and $X^*_{1:N}$ are a sample from the approximate smoother distribution $\hat{p}(X_{1:N}, \mathcal{T} \mid y_{1:N})$

index selection). If jumps (times and types) alone are required, without a sample of $X_{1:T}$, this can be achieved using the related Forward-Filter Backward Sampling algorithm of [183; 184], which produces approximate samples from the jump parameter distribution $p(\mathcal{T} \mid y_{1:N})$ without sampling the linear Gaussian portion of the state. The method of [183; 184] uses a similar forward-filtering, backward sampling idea to that shown in this section, but values of X_n are not sampled during the backward sampling pass. Instead, their linear Gaussian structure is exploited to marginalize out the jump parameters. The algorithm in [183; 184] can also be used in place of that shown in this section to draw samples for both jump parameters and $X_{1:N}$, by first drawing a sample of jumps $\mathcal{T}^*_{1:N}$ from their smoother distribution, and then backward sampling $X^*_{1:N}$, conditioned on these jumps using the backward sampling method in section 3.4.1.

3.5 Application: Trend-following in Finance

Multivariate linear jump diffusion models of the type presented above can be used to model momentum effects in financial assets, as shown in our earlier work in [2], upon which this section draws heavily.

Momentum strategies have been the source of much academic debate (e.g. [185; 186; 187; 188; 189]) because they appear to defy even the weak form of the *Efficient Market Hypothesis* (EMH) of [190], which states that prices should not be predictable from analysing their past history. This form of the EMH is consistent with random-walk behaviour of asset prices and suggests that no form of *technical analysis* (price prediction based solely on studying previous price history) can generate above average returns without taking above average risks [185]. However, technical analysis remains in widespread use in public markets [191] and various forms have been shown to have at least some predictive power [191; 192; 193; 194]. Momentum effects in particular have been extensively studied (e.g. [195; 187; 196; 197; 198; 199], amongst others) and have been found to exist in

a number of markets including foreign exchange [198], commodities [200] and equities [197]. Proposed explanations of these effects include the non-instantaneous reaction of the market to news events [195], meaning that the effects of events take place over several trading periods, and herding behaviour [201], in which, for example, investors clamour for assets that have recently performed well, further increasing their price. Though this herding behaviour has been termed "irrational exuberance" [201], irrationality of investors is not required to explain momentum effects [188; 202]. Some advocates of market efficiency contest that trading costs would wipe out any practical benefit of momentum trading [185; 189]. But, despite efficient market objections to momentum trading, the continuing existence and profitability of momentum-based funds has led to obvious ongoing interest. Momentum effects have been found at a range of frequencies from multi-monthly [187], to intraday [203], though this latter study found them to be more profitable at higher (intraday) frequencies. A range of methods have been used to attempt to find trends in price data, including some very simple strategies such as buying shares that performed well over a previous time period, which have been shown to be effective under certain circumstances [187].

Here, it is proposed that trend information be found by using model based tracking algorithms, using a variant of the jump diffusion model described above. As with all model based tracking applications the fit of the dynamical model to the true dynamics has a substantial effect on tracking performance [204]. However, the underlying dynamics of financial asset values are much less clear than in the physical case, despite extensive study of the behaviour of asset prices, e.g. [205; 206; 207; 208; 209]. Though the 'stylized facts' established in this literature can help point out shortcomings of an asset price model, they do not prescribe a specific model of asset dynamics. Numerous models have been proposed, largely dependent on the application and their analytical tractability; popular models include GARCH models in econometrics [210], exponential Brownian motion mod-

els in Black-Scholes option pricing [211], stochastic volatility models to fit volatility clustering and option price smiles [212] and jump diffusion and Lévy process models to fit heavy tails, e.g. [213] and [214], respectively. Such random walk models, however, do not allow for a predictable trend in asset prices. Since it is the aim of this model to determine such a trend (if one exists), the model used must introduce a 'trend' term. The model described here is similar in spirit to the "near constant velocity" physical tracking models described in [204]. It can also be viewed as an extension of the Langevin dynamics used in [172]. By allowing for jumps in the trend process, the models attempt to ameliorate the problem of changing trends, which can cause difficulty for momentum strategies [215], since at the point a trend changes, a momentum strategy following that trend can make significant losses if it is unable to identify the change quickly and alter its position appropriately.

3.5.1 A Model for Trend Following in Finance

The model considered here is a two-dimensional model consisting of a 'value' x_1 and 'trend' x_2 component. Both components are mean-reverting random processes subject to Gaussian noise of constant volatility and random jumps. Including mean reversion in the model reflects a view that trends are likely to fade over time. Figure 3.2 illustrates the types of changes in trend and price that can be accommodated by such a model. Observed prices y are modelled as observations of the value process x_1 subject to Gaussian noise. Noise in the state dynamics is modelled as being Gaussian and independent for each process, as is the distribution of jump sizes in each process. Using the notation introduced in section 3.2, the governing SDE for the state dynamics in this model is given by

$$\begin{bmatrix} dx_{1,t} \\ dx_{2,t} \end{bmatrix} = \begin{bmatrix} \theta_1 & 1 \\ 0 & \theta_2 \end{bmatrix} \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix} dt + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} dW_t + \begin{bmatrix} \sigma_{j_1} & 0 \\ 0 & \sigma_{j_2} \end{bmatrix} dJ_t, \quad (3.30)$$

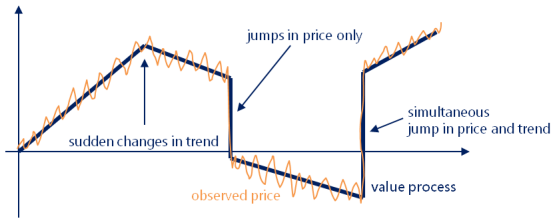


Figure 3.2: Types of jumps that can be modelled

which corresponds to the following values for the A, B and C matrices in equation (3.1):

$$X_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix}, \quad A = \begin{bmatrix} \theta_1 & 1 \\ 0 & \theta_2 \end{bmatrix}, \quad B = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}, \quad C = \begin{bmatrix} \sigma_{j_1} & 0 \\ 0 & \sigma_{j_2} \end{bmatrix}. \tag{3.31}$$

A simple observation model is assumed in which the i^{th} price observation, y_i, observed at time t_i, is the result of an observation of the value process x_{1,t_i}, perturbed by Gaussian noise of a fixed variance:

$$y_i = x_{1,t_i} + \nu_{t_i}, \tag{3.32}$$

where $\nu_{t_i} \sim \mathcal{N}(0, \sigma^2_{obs})$. This gives an observation density, conditional on the state at time t_i, of

$$y_i \sim \mathcal{N}(x_{1,t_i}, \sigma^2_{obs}). \tag{3.33}$$

and corresponds to observation matrix H and covariance Σ_{obs}

$$H = [1 \;\; 0], \quad \Sigma_{obs} = \sigma^2_{obs}. \tag{3.34}$$

Jumps in each process are modelled as being independent and following a Gauss-Poisson process, with jump times τ_k following a Poisson arrival process with rate λ_i, $i \in \{1, 2\}$, for each process, and jump sizes S_k

distributed as a multivariate Gaussian, so that

$$S_k \sim \mathcal{N}\left(0, \begin{bmatrix} \mathcal{I}_{\text{jump}_1(\tau_k)} & 0 \\ 0 & \mathcal{I}_{\text{jump}_2(\tau_k)} \end{bmatrix}\right) \tag{3.35}$$

$$\tau_k - \tau_{k-1} \sim \min\left(\text{exponential}\left(\lambda_1\right), \text{exponential}\left(\lambda_2\right)\right) \tag{3.36}$$

$$= \text{exponential}\left(\lambda_1 + \lambda_2\right) \tag{3.37}$$

with J_t being defined as in equation (3.11).

This model has nine parameters: the mean reversion coefficients θ_1 and θ_2, the diffusion noise variances σ_1^2 and σ_2^2, the jump rates λ_1 and λ_2, the jump size variances $\sigma_{J_1}^2$ and $\sigma_{J_2}^2$, and the observation noise variance σ_{obs}^2. It generalizes the model in [2] by introducing the possibility of random innovations (both jumps and diffusion) in the value process, and by allowing this process itself to be mean reverting, which could be of use when dealing with price movements in certain asset classes such as currency pairs. The model from [2] is easily recovered by setting the parameters relating to the x_1 process σ_1, θ_1, σ_{j_1} and λ_1 to 0, leaving the five parameters in [2]. This latter model is referred to as the *Langevin model*, owing to the governing SDE of its dynamical model resembling a Langevin equation. The model with jumps, diffusion and mean reversion in both processes is referred to at the *full model*.

3.6 Results

In order to demonstrate the operation of the VRPF filter it was applied to data generated from the two-factor model described in section 3.5.1. The results are shown in figure 3.3. The two sets of results show the estimates derived from the filter immediately after each observation, and at a lag of 5 observations. This latter is a particle approximation to the posterior fixed-lag smoother distribution $p(X_{t-L} \mid y_{1:t})$, where L is the lag in observation periods. It is given by the collection of fixed-lag smoother distributions $p(X_{t-L}^i \mid y_{1:t}, \tau^i)$ corresponding to each particle in the particle collection at

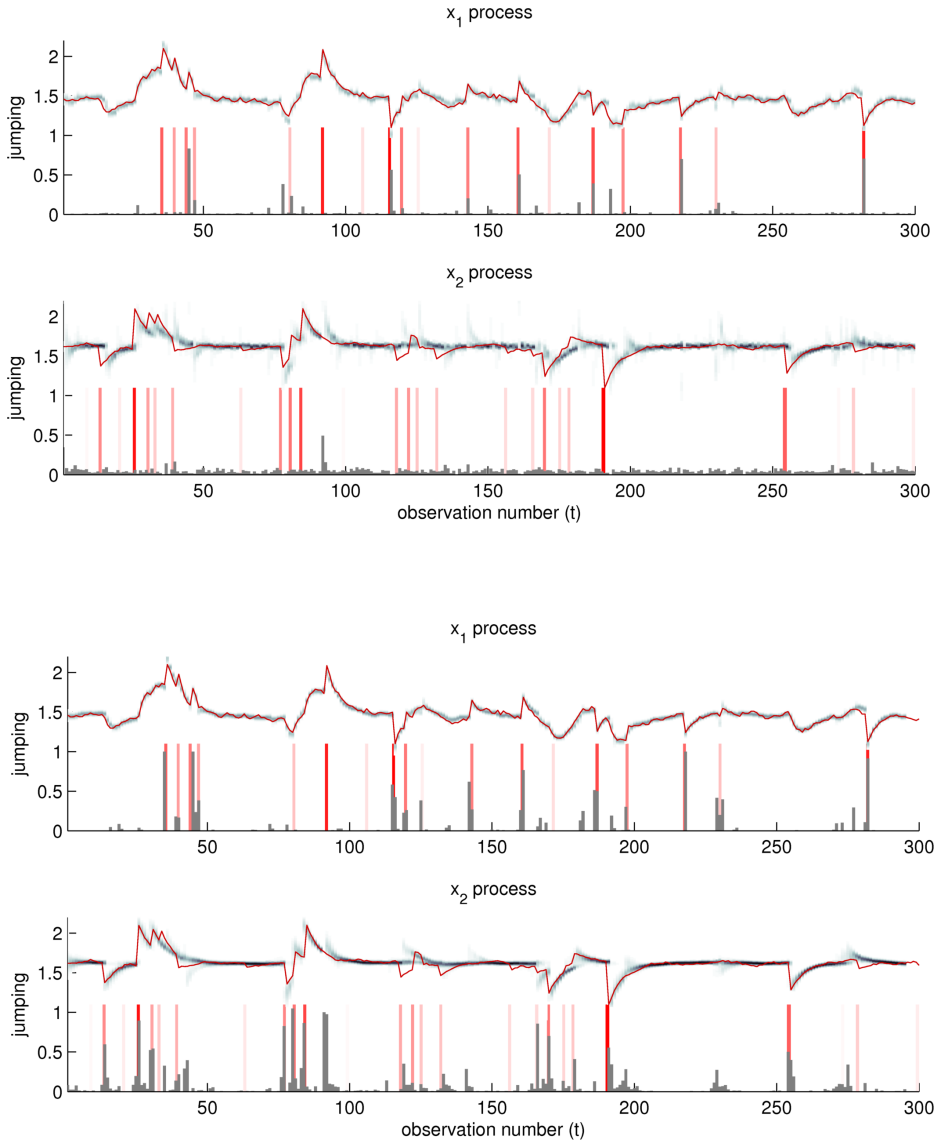


Figure 3.3: Typical VRPF filter output for two factor model. The true process is shown red and inferred posterior distribution is shown by grey shading (processes have been scaled to fit in range [1.2, 2.2]). Red bars indicate true jump positions with shading intensity representing jump size. Inferred jumps are shown by grey bars with height indicates total weight of particles containing a jump between observation times. Upper figure shows filter output, lower figure shows fixed lag smoother output at lag of 5 observations

time t, weighted by the particle weights, so that

$$\begin{aligned} p(X_{t-L} \mid y_{1:t}) &\approx \sum_i w_t^i p(X_{t-L}^i \mid y_{1:t}, \tau^i) \\ &= \sum_i w_t^i \mathcal{N}\left(X_{t-L}; \mu_{t-L|t}^i, \Sigma_{t-L|t}^i\right), \end{aligned}$$

where $\mu_{t-L|t}^i$ and $\Sigma_{t-L|t}^i$ are the mean and covariance of the process state obtained given the jump sequence of particle i from a fixed lag smoother of lag L. This smoother estimate can be provided by the Kalman filter itself by incorporating lagged states into the filter's state space or using a fixed-interval smoother such as the Rauch-Tung-Striebel smoother; see sections 2.1.2 and 2.1.2, respectively.

In figure 3.3 the fixed-lag approximation is significantly better than the immediate filter output, especially in terms of jump detection in the x_2 (trend) process, which is poor in the filter. The response to jumps is quicker and more accurate in the fixed-lag approximation, which can be seen by the fact that the state estimates regain the true signal more quickly after jumps (for example, this is visible after jumps in the x_2 process between $t = 150$ and 200).

Due to resampling, the quality of the fixed-lag approximation can be poor at long lags, since the particle history degenerates to a handful of common ancestor particles beyond a certain number of generations. Figure 3.4 shows the mean squared error per observation for a range of lags. It indicates that for this data the error falls rapidly to a lag of about 5, then levels off, increasing slightly at very long lags, with the variance of the estimates increasing for long lags. The expected squared estimate error at time t is defined as

$$\begin{aligned} \mathbb{E}\left(\text{error}^2\right) &= \int (\hat{x}_t - x_t)^2 p(\hat{x}_t) d\hat{x}_t, \\ &= \mathbb{E}\left(\hat{x}_t^2\right) - 2x_t \mathbb{E}\left(\hat{x}_t\right) + x_t^2, \end{aligned}$$

where $\hat{x}_t$ is the state estimate and x_t is the true system state at time t. Since

the output of the filter (i.e. the distribution $p(\hat{x})$) is given by a Gaussian mixture (with one component for each particle in the collection, corresponding to the Kalman filter distribution for that particle's particular set of jumps), this error is given by

$$\mathbb{E}\left(\text{error}^2\right) = \sum_i w_i(\sigma_i^2 + \mu_i^2 - 2x_t\mu_i) + x_t^2, \tag{3.38}$$

where w_i is the weight of particle i, and μ_i and σ_i^2 are the mean and variance of the posterior distribution corresponding to the jumps in particle i at time t. In order to calculate the mean error of the fixed lag estimates, the mean of this error is taken over all times for which the estimate is available (which will stop L periods before the final observation).

In some cases, it might be preferable to use the expected absolute error. For a Guassian mixture estimate $\hat{x} \sim \sum_i w_i \mathcal{N}\left(\mu_i, \sigma_i^2\right)$ of a quantity x, an analytic expression can be derived for this as follows.

$$\begin{aligned}
\mathbb{E}\left(|x - \hat{x}|\right) &= \int_{-\infty}^{x} (x - \hat{x}) p(\hat{x}) d\hat{x} + \int_{x}^{\infty} (\hat{x} - x) p(\hat{x}) d\hat{x} \\
&= \sum_i w_i \int_{-\infty}^{x} (x - \hat{x}) \mathcal{N}\left(\hat{x}; \mu_i, \sigma_i^2\right) d\hat{x} \\
&\quad + \sum_i w_i \int_{x}^{\infty} (\hat{x} - x) \mathcal{N}\left(\hat{x}; \mu_i, \sigma_i^2\right) d\hat{x} \\
&= \sum_i w_i (x - \mu_i)[2\Phi(x; \mu_i, \sigma_i^2) - 1] + 2\sigma_i^2 \mathcal{N}\left(x; \mu_i, \sigma_i^2\right),
\end{aligned}$$

where $\Phi(x; \mu, \sigma^2)$ is the Gaussian CDF at x for a Gaussian distribution with mean μ and variance σ^2.

3.6.1 Backward Sampling

To compare the results of backward sampling, samples were drawn from the smoother distribution $p(X_{1:T} \mid y_{1:T})$ for a range of series lengths and using various numbers of particles in the forward filter. Figure 3.5 shows the mean squared errors of each of these tests, including, as a reference

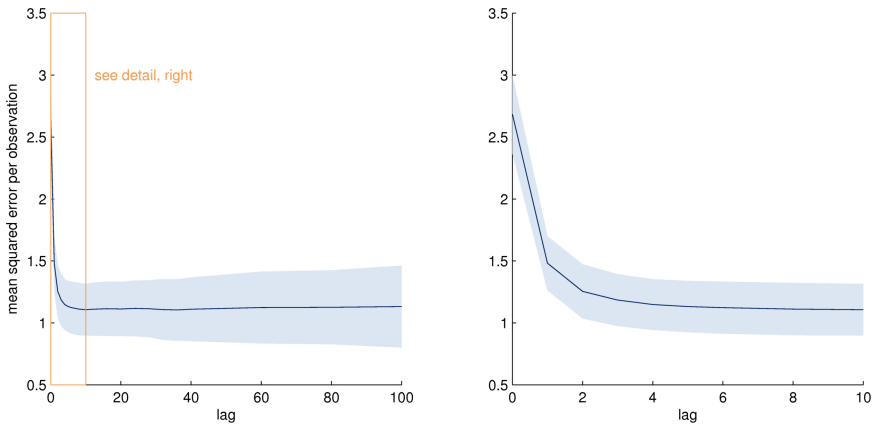


Figure 3.4: Mean squared fixed lag smoother error per observation by lag (a lag of 0 is the filter). Mean taken over 30 different data sets of length 300 observations. Shaded area shows $\pm$ one standard deviation. Right hand plot shows detail over first 10 lags

the expected mean squared error of the forward filter distributions calculated by equation (3.38). Both graphs show very similar mean squared errors with and without back sampling of jump times. This is surprising, since back sampling jumps might be expected to decrease the overall error, since samples should better approximate the smoother distribution. The result *could* be due to the final particle filter approximation to the smoother distribution being more sharply concentrated in a few areas of high likelihood, owing to the particle filter's resampling process keeping well-fitting particles and discarding others. This would lead to samples from the final particle distribution being of high likelihood and low error, but with a limited distribution that would not well reflect the true smoother distribution, especially in cases of multimodality. This situation is illustrated in figure 3.6, though it is not certain that this is what is occurring here. These results are consistent with those in [183; 184], in which a large improvement in sample diversity but only a "small improvement in [RMS] accuracy from the smoother" [184] was found when comparing results from the filter-smoother and forward-filtering backward sampling methods for linear Gaussian jump-diffusion systems.

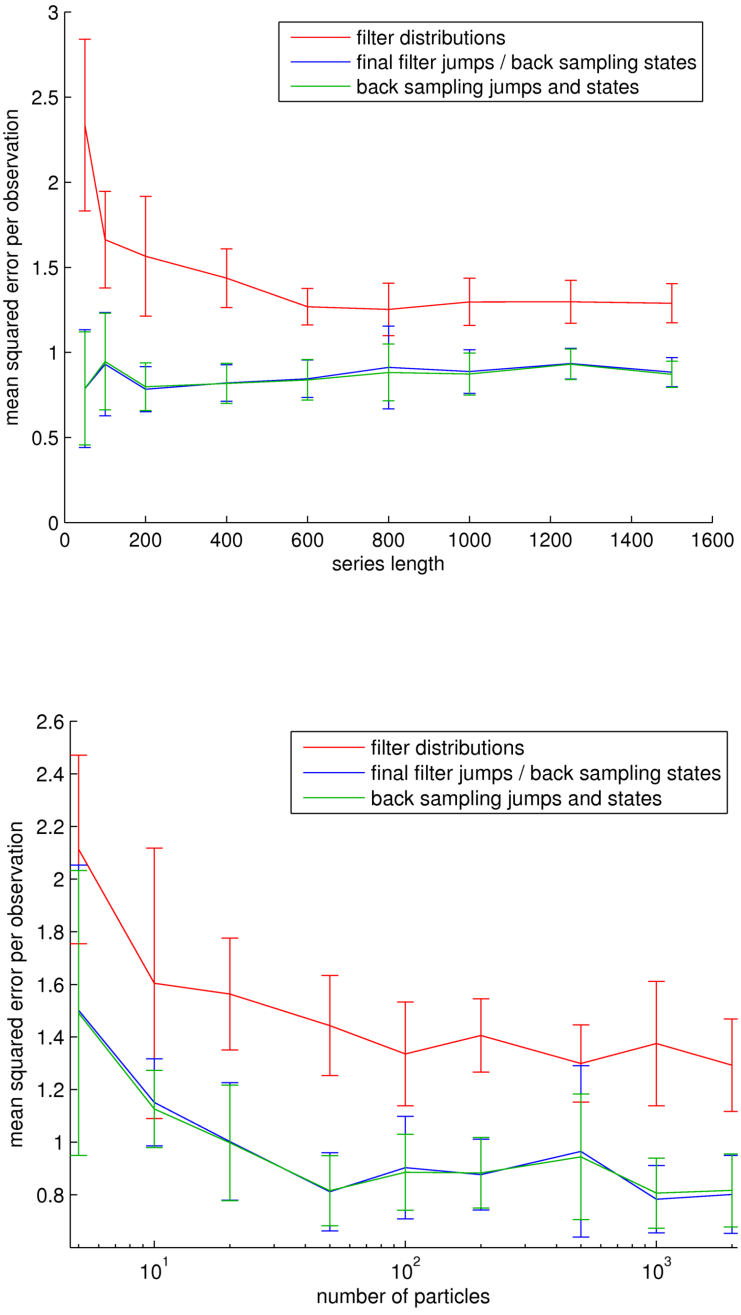


Figure 3.5: Mean squared state error per observation by series length (top) and number of particles (bottom) for a range of jump sampling methods. Error bars show one standard deviation. Series length test (top) run with 50 particles, 10 different sets of data per series length and 100 samples per set of data. Particle number test (bottom) run with series of 400 observation. Filter results show expected mean squared error as given in equation (3.38)

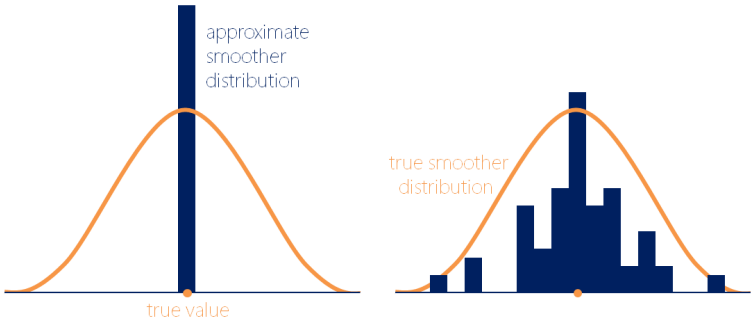


Figure 3.6: Two different approximations to the smoother distribution; the approximation on the left will give lower error to the true value but is a worse approximation to the true smoother distribution than that on the right

The tests with varying numbers of particles show that, as expected, increasing the number of particles generally decreases the error for both the back sampler and filter, although this effect is not especially pronounced beyond about 500 particles for this model and data. Series length does not have a significant effect on error, except for short series, where the uncertainty in the prior and thus error in initial samples is amortized over fewer observations.

The advantage of back sampling can be seen in figures 3.7 and 3.8, which show the results of sampling from the approximate smoothing distributions with and without backward sampling ('Back Sampling Jumps' and 'Final Particles Jumps', respectively). Figure 3.7 shows the positions of sampled jumps using each method. This illustrates the degeneracy of the samples drawn from the filter-smoother distributions; in early time periods there are perhaps two ancestor paths that represent the entire smoother distribution, although the location of the jumps found are generally good. In contrast, the back sampled jumps are diverse throughout the sample, clustering around the true jumps. The average number of jumps has also been found to be consistently closer to the true number of jumps using jump back sampling.

Figure 3.8 shows the distribution of jump time and process state samples

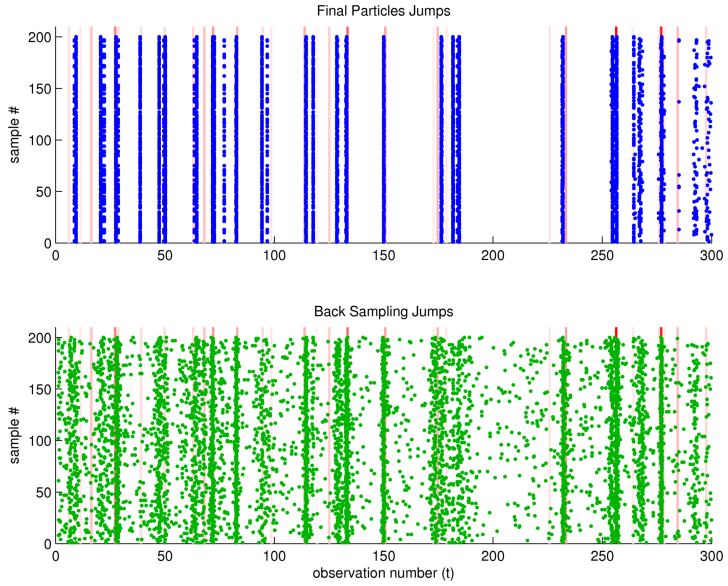


Figure 3.7: 200 samples of jump times drawn from the final particle filter (filter-smoother) distribution (top) and by using VRPF backward sampling (bottom); red bars show true jump times

for each of the two sampling methods. Samples drawn from the filter-smoother distributions have only a few jump times occurring in all samples, especially in early periods, whereas samples for which jump times were also back sampled have more diffuse jump time distributions, clustering about the true jump times. The advantage of back sampling jump times is also clear in the process state estimation, especially in early stages. For example, for the particle filter jumps, jumps in all samples at around observations 25, 80 and 240 lead to the true process lying outside three standard deviations of the samples, whereas for the back sampled jumps this is not the case. A couple of mis-sampled jumps between observations 140 and 150 lead to excessive variance in samples using filter-smoother jumps when compared to the back sampled jumps.

In this case, back sampling takes around one tenth the time of running the particle filter itself, so is a useful method for drawing a relatively small number of diverse samples from the approximate smoother distribution.

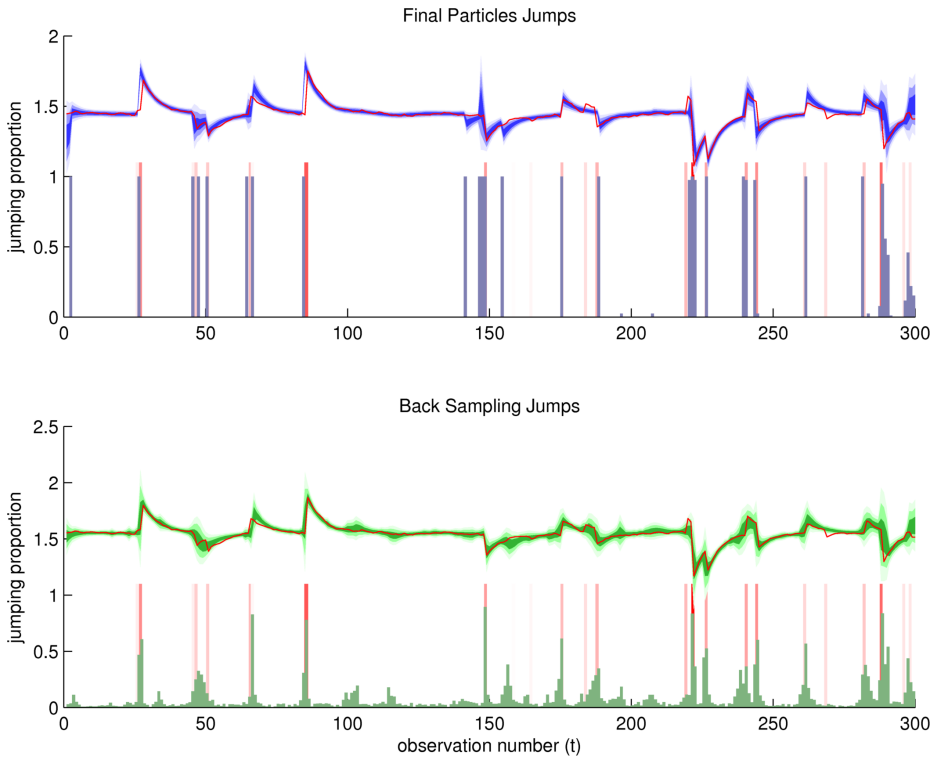


Figure 3.8: Distribution over 400 back samples for samples drawn without (top) and with (bottom) jump resampling. x_2 process only shown. Red line shows true x_2 process, shading shows 1 (darkest shading), 2 and 3 (lightest shading) standard deviations of back samples at each observation time. Sampled jump times are shown as proportion of back samples having a jump between t and $t+1$ for each integer observation time t, with true jumps marked by red bars

Parameter	Full Model	Langevin Model	Langevin Model in [2]
σ_1	5		
σ_2	4	4	4.1
σ_{j_1}	50		
σ_{j_2}	45	45	70
σ_{obs}	8	8	20
θ_1	0		
θ_2	-0.1	-0.1	-0.2
λ_1	0.05		
λ_2	0.05	0.05	0.2

Table 3.1: Parameter values used for the Full and Langevin models

3.6.2 Foreign Exchange Data

The model in section 3.5 was applied to five currency pairs over 14 years. Currency pairs were chosen because they might be expected to be mean reverting (at least to some extent), rather than having a pattern of long-term growth as might be seen with equities. In order to turn the output of the filter into a trading strategy a very simple approach was taken: if the mean predicted price ratio at $t+1$ was greater than the current price ratio at time t, the denominator currency was held for the period from t to $t+1$ (as it was predicted to become more valuable), otherwise the numerator currency was held from t to $t+1$. Much more sophisticated strategies could be developed, with the trading decision based on the predicted probability of an up or down move in the price ratio, as estimated by the filter. Figure 3.9 shows the result of applying this strategy to five currency pairs from March 1999 to March 2013. Both the full model and the Langevin model were tested with parameters given in table 3.1. These parameters were based on those used for the similar model in [2] and on inspection of the filter results, to ensure that these were correctly scaled for the data. In each one year period, each series was offset by its initial value so that it started at 0; series were also scaled by 1000 (this put them in a more familiar range for choosing parameters, but has no other effect on the results).

The results in figure 3.9 are somewhat positive for the four series in-

volving the US Dollar (making positive returns of between about 0.7% and 2% per year). However for the EUR/GBP series, the strategy consistently and steadily lost value, which cautions against too much optimism. Different parameter values might improve the performance here. In our earlier work [2], a similar model was found to produce positive returns across a portfolio consisting of a large basket of assets, although the filter output was processed in a somewhat different way to produce a trading strategy.

Figure 3.10 shows an example strategy for a single year of JPY/USD data. It nicely illustrates some of the successes and problems encountered when applying trend following algorithms to financial data. The algorithm performs well during the strong trends from day 55 to 90 and especially from day 210 to 250. Here the price ratio continues along a clear trend for a significant period of time, which allows the filter to pick up on the underlying trend and to profit from that. In contrast, in the period between about day 85 and day 200 there is no clear trend for the filter to follow, with the price oscillating somewhat; here the algorithm achieves very little and frequently gets the direction wrong. Finally, from day 50 to 60 the algorithm chooses to buy dollars, correctly following the prevailing trend. However, at the end of this buying period a nasty surprise awaited it, in the form of a large price fall. This more than wipes out the profit of following the earlier part of the trend and, although the algorithm detects and adapts quickly to the downward trend that ensues, it gets badly burnt at the point at which the trend changes. This is typical of trend following strategies; they must be sufficiently profitable in periods where trends continue to compensate for the inevitable losses when trends reverse. Overall, in this period the algorithm picked the correct direction in just 43% of cases, but made a small profit of around 0.2% (due to correctly following strong trends).

3.7 Conclusions

This chapter showed how the variable rate particle filter can be applied to give an effective and computationally efficient method to infer the state of

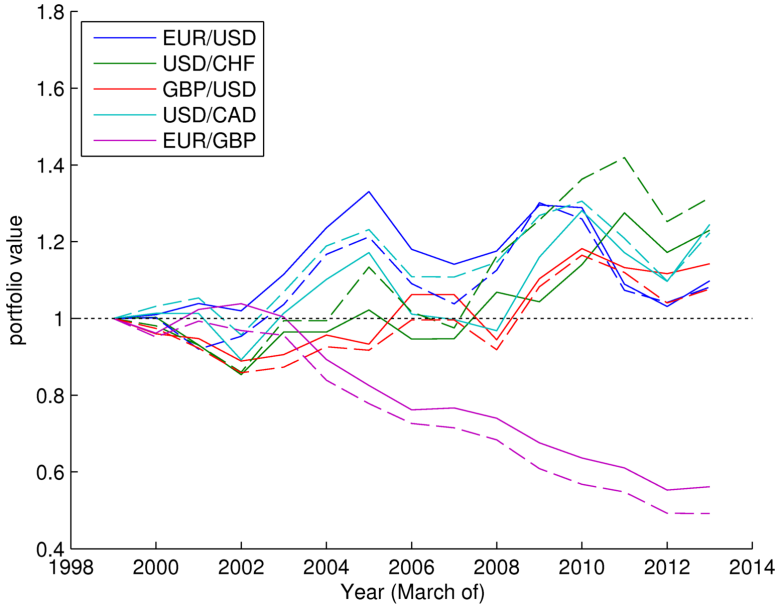


Figure 3.9: Portfolio return using a simple trading strategy based on the VRPF filter on five currency pairs from March 1999 to March 2013. Solid lines show results using full model; dashed lines show results using Langevin model

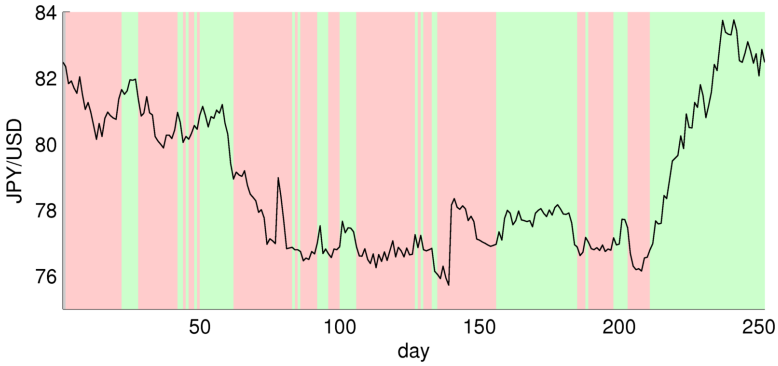


Figure 3.10: Example trading strategy derived using (full) model and VRPF filter. Data is JPY/USD from March 2011-March 2012 (increasing series value indicates increasing USD value). Green shading indicated a decision to buy USD, red shading indicates a decision to buy JPY

linear jump diffusion processes, given noisy observations.

The results in this chapter show that back sampling the jump times as well as the process state can significantly increase the diversity in the particle samples, giving samples that better represent the process state and jump times (particularly the uncertainty within the latter). This is particularly noticeable in early periods of the filter as particle diversity frequently declines to a very few common ancestors at long lags with computationally reasonable numbers of particles. The back sampling methods presented here are particularly useful as they can be used within the Particle MCMC parameter estimation described in chapter 4.

Though the financial results of applying these methods to follow trends in foreign exchange data were, at best, mixed, they illustrate the application of the algorithm in a concrete setting and do manage to identify and follow trends in the data when such trends exist. Financially they suffered from problems common to trend following algorithms, such as difficulty in periods with no clear trend, and suffering large losses when trends suddenly reversed.

Chapter 4

Parameter Inference for Linear Jump Diffusion Models

This chapter presents a number of methods for Bayesian parameter estimation in linear jump diffusion models of the type described in chapter 3. This is practically useful because such models have a wide range of applications (see chapter 3) and in many practical cases the choice of parameters for the models is far from intuitive. For example, the trend following model described in section 3.5 requires nine parameters to specify, the choice of which is not obvious either from knowledge of the asset types or from inspection of their previous behaviour. Estimation in a principled way from past data is therefore desirable.

Most existing parameter estimation methods for jump diffusion processes have focussed on directly observed jump diffusions rather than ones observed only via noisy observation. This obviates the need for simultaneous state and parameter estimation, since states are assumed known. Parameter estimation in such models has been tackled primarily using maximum likelihood (MLE) methods e.g. [216; 217; 169; 218] though other methods such as the method of moments [219] and least-squares fitting [220] have also been used. [174] suggest the use of particle filter likelihood maximization approaches, although these are prone to difficulties (see sec-

tion 2.3.4). The Bayesian estimation proposed here has the advantage of providing distributional parameter estimates as opposed to the point estimates obtained from MLE methods.

When presented with a batch of observations from which to estimate parameters, both the model parameters θ and the underlying system state and jumps, X and $\mathcal{T}$, are unknown and must be estimated. To do so, Gibbs sampling is employed, which allows samples to be drawn from the joint distribution $p(X, \mathcal{T}, \theta \mid y)$. Several Gibbs sampling schemes can be used for the parameters, relying on less or more marginalization of the state process, leading to traditional or collapsed samplers [221].

Two methods for drawing samples of the jump sequence $\mathcal{T}$ are presented here: reversible jump Markov chain Monte Carlo (RJMCMC) [34], or one of two Particle MCMC methods [33]. A modified version of the variable rate particle filter from chapter 3 is shown to be compatible with these latter methods. Both reversible jump and Particle MCMC methods allow asymptotically exact samples to be drawn from the joint jump and process state posterior distribution, in contrast to the approximate samples produced by the particle filter and backward sampling methods of the previous chapter. However, the estimation methods in this chapter are offline estimation methods suited for batch estimation rather than sequential inference.

The rest of this chapter is structured as follows. Section 4.1 gives details of the Gibbs samplers that can be used for parameter estimation. Section 4.2 outlines the RJMCMC scheme that can be used for state estimation. Section 4.3 introduces Particle MCMC methods and gives details of the specific schemes that can be used for jump diffusion models. Section 4.5 gives the results of a number of tests comparing the various methods proposed for parameter and state estimation. Finally, section 4.6 draws some conclusions from this work.

4.1 Gibbs Sampler for Parameters

The simplest Gibbs sampling approach, which can be applied to any parameter, is the Metropolis-within-Gibbs scheme. In this, a parameter θ_i is sampled from its conditional distribution $p(\theta_i \mid \theta_{-i}, \mathcal{T}, y)$ using

$$p(\theta_i \mid \theta_{-i}, \mathcal{T}, y) \propto p(y \mid \theta, \mathcal{T}) p(\mathcal{T} \mid \theta) p(\theta_i \mid \theta_{-i}). \tag{4.1}$$

The conditional likelihood of the observations $p(y \mid \theta, \mathcal{T})$ can be evaluated using the PED from the Kalman filter as described in section 2.1.2. The conditional likelihood of the jump sample $p(\mathcal{T} \mid \theta)$ can be evaluated from the jump transition density, since

$$p(\mathcal{T} \mid \theta) = p(\mathcal{T}_1 \mid \theta) \prod_{j=2}^{|\mathcal{T}|} p(\mathcal{T}_j \mid \mathcal{T}_{j-1}, \theta).$$

The distribution in equation (4.1) is not, in general, easy to sample. Sampling can be performed for each parameter θ_i using a Metropolis-Hastings step, with proposal $q(\theta_i^* \mid \theta_i')$, where θ_i' is the current sample and θ_i^* is the proposal, and acceptance probability

$$p_{\text{accept}} = \frac{p(y \mid \theta_{-i}, \theta_i^*, \mathcal{T}) p(\mathcal{T} \mid \theta_{-i}, \theta_i^*) p(\theta_i^* \mid \theta_{-i})}{p(y \mid \theta_{-i}, \theta_i', \mathcal{T}) p(\mathcal{T} \mid \theta_{-i}, \theta_i') p(\theta_i' \mid \theta_{-i})} \frac{q(\theta_i' \mid \theta_i^*)}{q(\theta_i^* \mid \theta_i')}.$$

This sampling mechanism is general, but is slow, because it requires evaluation of the likelihood, and can be inefficient if the proposal distribution is not well matched to the target distribution, leading to high rejection rates and poor mixing. In the absence of other information, a simple symmetrical Gaussian random walk proposal is used with variance chosen to match the scale over which a particular parameter is expected to vary.

4.1.1 Jump Rates

The jump rates $\lambda_{\{1,2\}}$ can be sampled efficiently if appropriate conjugate priors are chosen, since the inter-jump time for the x_1 and x_2 processes

are modelled as being independent and exponential with rate λ_i. In this case a Gamma$(\alpha_{\lambda_i}, \beta_{\lambda_i})$ prior on λ_i is conjugate and leads to the posterior distribution

$$p(\lambda_i \mid \mathcal{T}, y, \theta_{-\lambda_i}) = \text{Gamma}(\alpha_{\lambda_i} + n_i, \beta_{\lambda_i} + T),$$

where n_i is the number of jumps in process x_i and T is the total observed time of the process (i.e. $t_{\text{end}} - t_{\text{start}}$). This distribution can easily be sampled, leading to an efficient Gibbs sampler for the jump rates. It differs from the standard posterior distribution for the exponential rate parameter given n i.i.d. observations of the process because, in addition to the jumps that do occur, it must also take into account the fact that a jump does *not* occur between the final jump and the final observation, which also conveys some information about the jump rate (especially when there are no jumps).

For the avoidance of confusion, the Gamma distribution here is defined as

$$\text{Gamma}(\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\lambda\beta),$$

whereas some definitions (including that of the gampdf function in Matlab) use $1/\beta$ as the second parameter.

The prior parameters can be interpreted (in a sense) as effectively 'adding' $\alpha_{\lambda_i} - 1$ additional jumps to the jump sequence and 'adding' β_{λ_i} extra time units to the observation period when compared to the likelihood distribution for λ_i, which is given by $L(\lambda_i) = \text{Gamma}(n_i + 1, T)$. Figure 4.1 shows the effect on the posterior of an increasing number of jumps observed in a 100 time unit sequence; increasing the α prior parameter effectively moves this posterior a number of steps to the right.

4.1.2 Sampled State

For some parameters an alternative to the Metropolis-within-Gibbs scheme above is to sample the system state X and use this sample in order to sample

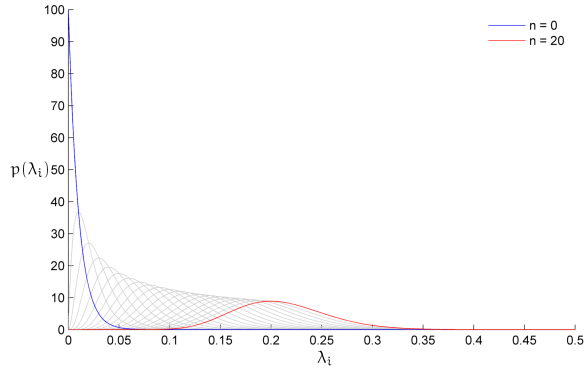


Figure 4.1: Posterior distribution of λ_i for values of n_i ranging from 0 (blue) to 20 (red), for $\alpha = 1$, $\beta = 1$ and $T = 100$

the relevant parameters. Doing this allows the jump variance $\sigma^2_{J_{\{1,2\}}}$ for each process, and the the observation noise variance σ^2_{obs} to be estimated efficiently given suitable conjugate priors.

According to the observation model, observations are equal to the value of the x_1 process perturbed by zero-mean independent Gaussian noise with a constant variance σ^2_{obs}. Thus, given a sample of x_1 and the observations, the difference between them can be used to infer the observation noise. In this case, an inverse gamma $\mathcal{IG}(\alpha_{obs}, \beta_{obs})$ prior distribution on σ^2_{obs} is a conjugate prior, leading to the easy-to-sample posterior distribution

$$p(\sigma^2_{obs} \mid X, y) = \mathcal{IG}\left(\alpha_{obs} + \frac{N}{2}, \beta_{obs} + \frac{1}{2}\sum_{n=1}^{N}(x_{1,t_n} - y_n)^2\right),$$

where N is the total number of observations.

Similarly the jump variance $\sigma^2_{J_{\{1,2\}}}$ can be inferred given the system state before and after a jump (which can be obtained from the backward sampling algorithm in section 3.4.1). According to the model, jumps sizes are independent and normally distributed with zero mean, thus an inverse gamma $\mathcal{IG}(\alpha_{J_i}, \beta_{J_i})$ prior distribution on $\sigma^2_{J_i}$ leads to the posterior distri-

bution

$$p(\sigma^2_{J_i} \mid \mathcal{T}, X, y) = \mathcal{IG}\left(\alpha_{J_i} + \frac{|J_i|}{2}, \beta_{J_i} + \frac{1}{2}\sum_{j=1}^{|J_i|}(x^+_{i,\tau_j} - x^-_{i,\tau_j})^2\right),$$

where $|J_i|$ is the number of jumps in the x_i process and x^-_{i,τ_j} and x^+_{i,τ_j} are the values of the process x_i before and after the jump τ_j.

Other parameters can also be estimated given a sample of X using a Metropolis-within-Gibbs step similar to that employed when sampling from equation (4.1) but replacing $p(y \mid \mathcal{T}, \theta)$ with $p(X \mid \mathcal{T}, \theta)$. However, sampling the state and using the sample to evaluate likelihood introduces an unnecessary 'sampling noise' into the likelihood being considered. This can be significant in areas where the likelihood is relatively flat, since this 'noise' can be at a much larger scale than the variation in the likelihood itself, and can lead to poor results.

4.1.3 Hybrid Scheme

In order to accommodate that some parameters can be efficiently sampled after sampling X (call these θ_X), whilst others are best sampled with X marginalized out (call these θ_m), a sampling scheme incorporating both types of sampling can be devised as follows.

1. Sample $X, \mathcal{T}, \theta_m \sim p(X, \mathcal{T}, \theta_m \mid y, \theta_X)$
   (a) Sample $\mathcal{T} \sim p(\mathcal{T} \mid \theta_m, \theta_X, y)$ by RJ-MCMC or Particle Gibbs (see sections 4.2 and 4.3.4)
   (b) Sample $\theta_m \sim p(\theta_m \mid \mathcal{T}, \theta_X, y)$ by Metropolis-within-Gibbs (see above)
   (c) Sample $X \sim p(X \mid \theta_m, \theta_X, \mathcal{T}, y)$ by backward sampling (see section 3.4.1)
2. Sample $\theta_X \sim p(\theta_X \mid X, \theta_m, \tau, y)$

In this scheme, steps 1a and 1b form a collapsed Gibbs sampler [221] for $\mathcal{T}$ and θ_m. After these are sampled, a sample can be drawn for the hidden state X via backward sampling. This sample can then be used in step 2 to sample the parameters that are best sampled conditional on X. The validity of this scheme can also be seen by viewing Steps 1a and 1b as drawing samples from $p(\mathcal{T}, [X] \mid \theta_m, \theta_X, y)$ and $p(\theta_m, [X] \mid \mathcal{T}, \theta_X, y)$, respectively, where [X] denotes a sample of X that is discarded and so in practice need never be sampled. This is valid because these samples of X are never used in sampling any other variables.

4.2 Sampling Jumps: Reversible Jump MCMC

Sampling from the jump distribution $p(\mathcal{T} \mid y, \theta)$ can be achieved using reversible jump MCMC [34]. The state of the chain consists of the entire jump sequence $\mathcal{T}$ and therefore proposals must be such that a series of accepted proposals is able to transform any jump sequence into any other possible jump sequence.

To this end three simple proposal types are allowed: a *move* proposal, in which one jump time (and possibly type) is altered locally; a *birth* proposal, in which a new jump is created; and a *death* proposal, in which an existing jump is removed. These are shown in figure 4.2 and allow any starting sequence of jumps to be transformed to any other through a series of moves, births and deaths.

Because the dimension of the state space can change (with birth and death proposals), proposals are actually a map between one state space and the random variables used to generate the proposal, and another state space (of possibly different dimension), along with the random variables used to generate a reverse proposal. Together, the state/random variable sets have the same dimension and normal MCMC (with a change of vari-

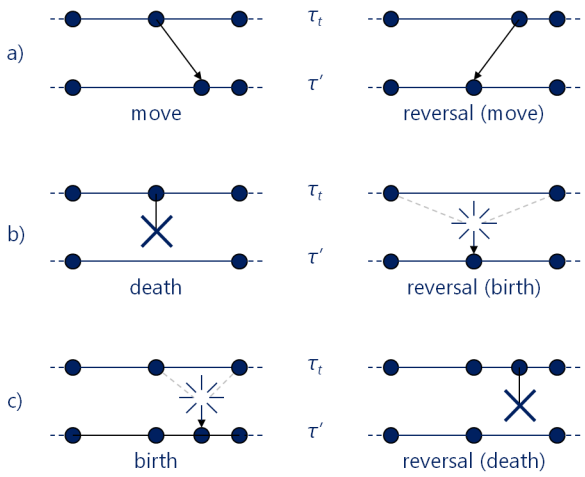


Figure 4.2: The three basic types of proposal for state sequence updates, along with their reversals: (a) move, (b) death and (c) birth

ables) can be applied. In this case the acceptance ratio is given by [34]

$$\alpha(\mathcal{T}, \mathcal{T}') = \min\left(1, \frac{p(\mathcal{T}' \mid y, \theta) g'(u')}{p(\mathcal{T} \mid y, \theta) g(u)} \left| \frac{\partial(\mathcal{T}', u')}{\partial(\mathcal{T}, u)} \right| \right) \tag{4.2}$$

where u is the set of random variables necessary to propose $\mathcal{T}'$ from $\mathcal{T}$ (and *vice versa* for u'), and $g(u)$ is the density from which these are proposed. In order for the dimensionality to balance, it is required that

$$\dim(\mathcal{T}) + \dim(u) = \dim(\mathcal{T}') + \dim(u'). \tag{4.3}$$

The proposals from $(\mathcal{T}, u)$ to $(\mathcal{T}', u')$ take the form of deterministic functions of u and $\mathcal{T}$, so that $h(\mathcal{T}, u) = (\mathcal{T}', u')$. These must be reversible, so that $h^{-1}(\mathcal{T}', u') = (\mathcal{T}, u)$ exists and, due to the appearance of the derivative in equation (4.2), must be differentiable, since $\frac{\partial(\mathcal{T}', u')}{\partial(\mathcal{T}, u)} = \frac{\partial h(\mathcal{T}, u)}{\partial(\mathcal{T}, u)}$. See [34; 105; 222] for full details and proof that this satisfies detailed balance.

For example, a birth proposal maps the current set of N jumps $\mathcal{T} \in (\mathbb{R} \times \mathbb{T})^N$ (where $\mathbb{T}$ is the space of jump types, and assuming that jump times are in $\mathbb{R}$), along with a new jump proposal consisting of a time, type and place in the sequence $u \in \mathbb{R} \times \mathbb{T} \times \mathbb{Z}_+$ to a new space $\mathcal{T}' \in (\mathbb{R} \times \mathbb{T})^{N+1}$

and the place of the jump in the new sequence $u' \in \mathbb{Z}_+$. Thus, for $u = (J, i)$ where J is a new jump (time and type) and i is the index of the latest earlier jump the existing sequence (or 0 if there are none), the birth map is

$$h_{\text{birth}}(\mathcal{T}, (J, i)) = \left(\begin{bmatrix} \mathcal{T}_{1:i} \\ J \\ \mathcal{T}_{i+1:N} \end{bmatrix}, i+1 \right). \tag{4.4}$$

Here $u' = i + 1$ and i is deterministic given the time of J and the current sequence $\mathcal{T}$. The reversal of this proposal is

$$h^{-1}_{\text{birth}}(\mathcal{T}', u') = \left(\begin{bmatrix} \mathcal{T}'_{1:u'-1} \\ \mathcal{T}'_{u'+1:N'} \end{bmatrix}, \left(\mathcal{T}'_{u'}, u' - 1\right) \right),$$

where N' is the number of jumps in $\mathcal{T}'$ (i.e $N + 1$). This can be seen intuitively as a death proposal, since it removes the jump in the u'^{th} position in the sequence. Thus, birth and death proposals can really be seen as a single proposal type.

Since equation (4.4) can be rewritten as

$$h_{\text{birth}}(\mathcal{T}, (J, i)) = \left(P \begin{bmatrix} \mathcal{T} \\ J \end{bmatrix}, i+1 \right)$$

where P is a permutation matrix that gives $\mathcal{T}'$ in the correct order, the Jacobian of the birth map is given by

$$\frac{\partial h_{\text{birth}}(\mathcal{T}, u)}{\partial(\mathcal{T}, u)} = \begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix},$$

and its determinant is therefore 1. In this case, the acceptance ratio in equation (4.2) simplifies to one that closely resembles the standard MCMC acceptance ratio. A similar result holds in the reverse (death) direction.

The generation of the random quantities for the birth proposal consists of generating a jump (time and type); the index is a deterministic function

of these. Since

$$g_{\text{birth}}(J, i \mid \mathcal{T}) = g_i(i \mid J, \mathcal{T}) g_{J_t}(J_{\text{time}} \mid \mathcal{T}, J_{\text{type}}) g_{J_x}(J_{\text{type}} \mid \mathcal{T}),$$

a possible g_{birth} is given by

$$\begin{aligned} g_i(i \mid J, \mathcal{T}) &= \delta_{\{\text{argmax}_i(\mathcal{T}^+_{i,\text{time}} < J_{\text{time}})\}} \\ g_{J_t}(J_{\text{time}} \mid \mathcal{T}, J_{\text{type}}) &= \mathcal{U}(t_0, t_{\text{max}}) \\ g_{J_x}(J_{\text{type}} \mid \mathcal{T}) &= \mathcal{U}(\mathbb{T}), \end{aligned} \tag{4.5}$$

where $\mathcal{T}^+ i, \text{time}$ is the set of jump times augmented with $\mathcal{T}^+_{0,\text{time}} = t_0$ and $\mathcal{T}^+_{N+1,\text{time}} = t_{\text{max}}$, the initial and final possible jump times, and $\mathcal{U}(\mathbb{T})$ is a uniform distribution over jump types.

In the opposite direction, the random generation is simply a matter of choosing the index of a jump to kill and so the simple uniform proposal can be used:

$$g'_{\text{birth}}(i' \mid \mathcal{T}') = \frac{1}{N'} \sum_{j=1}^{N'} \delta_{\{j\}}. \tag{4.6}$$

Using these uniform g for generation of the random elements of the proposals, the following straightforward proposals and acceptance ratios can be used.

Birth Birth proposals involve generating a new jump J (and index i) from $g_{\text{birth}}(J, i \mid \mathcal{T})$, as specified by the proposals from (4.5), above. This jump can then be added into the jump set as shown in equation (4.4), to create a jump set proposal $\mathcal{T}'$. This proposal can then be accepted with probability

$$\begin{aligned} \alpha_{\text{birth}}(\mathcal{T}, \mathcal{T}') &= \min\left(1, \frac{p(y \mid \mathcal{T}', \theta) p(\mathcal{T}') g'_{\text{birth}}(i')}{p(y \mid \mathcal{T}, \theta) p(\mathcal{T}) g_{\text{birth}}(J, i)}\right) \\ &= \min\left(1, \frac{p(y \mid \mathcal{T}', \theta) p(\mathcal{T}') (t_{\text{max}} - t_0) |\mathbb{T}|}{p(y \mid \mathcal{T}, \theta) p(\mathcal{T}) (N+1)}\right), \end{aligned}$$

where N is the number of jumps in $\mathcal{T}$ and $|\mathbb{T}|$ is the number of different types of possible jumps.

Death Death proposals are simply the reverse of birth proposals and can be created by choosing a jump J (index i) to kill using equation (4.6). By removing this jump from the current jump sequence, a proposal jump sequence $\mathcal{T}'$ is generated. By noting that the death proposal is simply the reverse of a birth proposal, this can be accepted with probability

$$\begin{aligned}\alpha_{\text{death}}(\mathcal{T},\mathcal{T}') &= \min\left(1, \frac{p(y \mid \mathcal{T}',\theta)p(\mathcal{T}')g_{\text{birth}}(J',i')}{p(y \mid \mathcal{T},\theta)p(\mathcal{T})g'_{\text{birth}}(i)}\right)\\ &= \min\left(1, \frac{p(y \mid \mathcal{T}',\theta)p(\mathcal{T}')N}{p(y \mid \mathcal{T},\theta)p(\mathcal{T})(t_{\max}-t_0)|\mathbb{T}|}\right),\end{aligned}$$

Move Move proposals do not change the state dimension and so can be treated as standard MCMC proposals. The strategy for creating move proposals used here is to randomly choose a jump i to move from the existing jump sequence. The time of this jump τ is then moved. Thus the random generation for this proposal can be expressed as

$$g_{\text{move}}(\tau, i \mid \mathcal{T}) = g_\tau(\tau \mid i, \mathcal{T})g_i(i \mid \mathcal{T}),$$

So that the proposed new jump sequence $\mathcal{T}'$ has its i^{th} jump replaced with one of the same type, but at time τ (the map that does this is straightforward). A simple move proposal can be created using a uniform random selection of the particle to move and a truncated Gaussian random addition to the current jump time (truncated so that the new jump does not move beyond the current neighbouring jump times or the ends of the possible interval). This gives

$$\begin{aligned}g_i(i \mid \mathcal{T}) &= \sum_{i=1}^{N} \delta_{\{i\}}\\ g_\tau(\tau \mid i, \mathcal{T}) &= \mathcal{N}^*\left(\tau\,; \mathcal{T}_i^{\text{time}}, \sigma^2_{\text{move}}\right),\end{aligned}$$

where $\mathcal{T}_i^{\text{time}}$ is the time of the i^{th} jump in $\mathcal{T}$, and $\mathcal{N}^*$ indicates a Gaussian distribution truncated at the relevant times.

This proposal $\mathcal{T}'$ can then be accepted with probability

$$\alpha_{\text{move}}(\mathcal{T}, \mathcal{T}') = \min\left(1, \frac{p(y \mid \mathcal{T}', \theta)p(\mathcal{T}')}{p(y \mid \mathcal{T}, \theta)p(\mathcal{T})}\right),$$

which can be seen because both the original and proposed jump time will be within the non-truncated region of the proposal distribution and so the proposal is effectively symmetric.

It is also possible, but not necessary, for move proposals to propose a change of jump type, but that is not considered here.

In all of the above cases, the likelihood $p(y \mid \mathcal{T})$ can be calculated using the PED of the Kalman filter (see section 2.1.2) and the prior $p(\mathcal{T})$ can be calculated from the jump model. Storage of intermediate state distributions allows this calculation to be performed more efficiently, since the likelihood need only be evaluated from the position of the jump preceding the selected one onwards (since no part of the likelihood calculation is affected for observations before that point). This would be expected to roughly double the speed of acceptance probability evaluation, since the selected jumps are uniformly distributed throughout the sequence.

4.3 Particle MCMC methods

Particle MCMC (PMCMC) methods were recently proposed in [33] as a way of using the approximate particle filter to draw asymptotically exact samples from distributions of interest. In particular, they can be used for exact sampling of hidden parameter and state variables in state space models. They rely on the particle filter's ability to calculate an estimate of the system likelihood and use this to calculate (exact) acceptance probabilities for the proposals generated. In this way they can be seen as a form of pseudo-marginal MCMC method, as discussed in section 2.2.3. They work

by constructing an extended target distribution over all the variables generated by the particle filter (i.e. all particle states and ancestor indices), which has the required posterior distribution as a marginal distribution. Since obtaining samples from marginals is straightforward given a set of joint samples, this allows samples to be drawn from the required posterior by discarding other auxiliary variables.

Three PMCMC methods are proposed in [33], the Particle Independence Metropolis-Hastings (PIMH) sampler, the Particle Marginal Metropolis Hastings (PMMH) sampler, and the Particle Gibbs (PGibbs) sampler. The first of these, PIMH, allows exact samples to be drawn from the posterior distribution $p(x_{0:T} \mid y_{1:T})$ of a state sequence $x_{0:T}$ given a series of observations $y_{1:T}$ (this notation is used throughout this section). PMMH allows draws to be made from the joint distribution of the state sequence and model parameters θ (or subsets thereof), $p(x_{0:T}, \theta \mid y_{1:T})$. Blocking strategies may be employed on the parameters θ in order to increase the probability of proposal acceptance, resulting in a Metropolis-within-Gibbs scheme, using the particle filter to evaluate acceptance probabilities. The PIMH and PMMH methods are both Metropolis-Hastings methods in which a proposal is generated via the particle filter (and a separate proposal mechanism for parameters) and accepted according to an acceptance probability defined in terms of tractable quantities derived from the particle filter. The final method, Particle Gibbs (PGibbs) is different in character, in that it targets the extended target distribution using a Gibbs sampler, in which samples are drawn using a series of draws from conditional distributions; a slightly modified version of the particle filter facilitates this for the state sequence.

The following sections outline the PMCMC methods and, in section 4.4, shows that the VRPF algorithm of section 3.3.1 can be adapted to work within them allowing exact samples to be drawn from $p(\mathcal{T}, \theta \mid y)$.

4.3.1 Particle Filter Algorithm

In order understand the Particle MCMC algorithms, it is necessary to make a precise statement of the particle filter algorithm used within them. The following notation is used throughout this section.

- $x_t = \{x_t^i \mid i = 1, ..., N_t\}$ is the particle collection at time t, where x_t^i is the state of particle i
- $w_t = \{w_t^i \mid i = 1, ..., N_t\}$ is the set of *un-normalized* particle weights at time t, where w_t^i is the un-normalized weight of particle i
- $v_t = \{v_t^i \mid i = 1, ..., N_t\}$ is the set of normalized particle weights at time t, where v_t^i is the weight of particle i
- $\tilde{x}_t = \{(x_t^i, v_t^i) \mid i = 1, ..., N_t\}$ is the collection of weighted particles at time t, which can be interpreted as the probability distribution $\sum_i^{N_t} v_t^i \delta_{\{x_t^i\}}$
- a_t^i is the ancestor (parent) of particle i at time t. This arises during the resampling stage of the particle filter, when the state history of each new particle at time t is drawn from the previous set of particles at $t-1$; the particle that supplies the history for each given new sample is its ancestor

Initialization: draw initial particles from prior; for all $i = 1, ..., N_0$

$$x_0^i \sim p(x_0),$$
$$v_0^i = 1/N_0.$$

Begin update step (t to $t+1$): assume that the particle collection $\tilde{x}_t$ is a collection of samples from the approximate filter distribution $\hat{p}(x_t \mid y_{1:t})$. This 'incoming' collection can contain any number of particles.

Resample: for each particle $i = 1, ..., N_t$ in the new generation, choose a

parent particle a_{t+1}^i from the resampling distribution

$$a_{t+1}^i \sim R(a_{t+1}^i \mid v_t).$$

The resampling function R must have the same support as the incoming particle collection (i.e. there must be a non-zero chance of choosing any non-zero weighted incoming particle).

Propagate: sample a new particle state from a proposal distribution for each particle:

$$x_{t+1}^i \sim q(x_{t+1}^i \mid x_t^{a_{t+1}^i}, y_{1:t+1}).$$

Weight: calculate an un-normalized weight w_{t+1}^i for each particle in light of the observation at $t+1$ as

$$w_{t+1}^i = \frac{p(y_{t+1} \mid x_{t+1}^i)p(x_{t+1}^i \mid x_t^{a_{t+1}^i})v_t^i}{q(x_{t+1}^i \mid x_t^{a_{t+1}^i}, y_{1:t+1})R(a_{t+1}^i \mid v_t)}. \tag{4.7}$$

In the simplest case of the bootstrap filter with multinomial resampling (such that $q(x_{t+1}^i \mid x_t^{a_{t+1}^i}, y_{1:t+1}) = p(x_{t+1}^i \mid x_t^{a_{t+1}^i})$ and $R(a_{t+1}^i \mid v_t) = v_t^i$), this weight becomes

$$w_{t+1}^i = p(y_{t+1} \mid x_{t+1}^i).$$

Note that w_0^i is not defined.

Normalize weights: normalize the particle weights so that they sum to 1.

$$v_{t+1}^i = \frac{w_{t+1}^i}{\sum_{i=1}^{N_{t+1}} w_{t+1}^i}.$$

This gives a particle collection $\tilde{x}_{t+1}$ that is an approximation of the posterior filtering distribution at $t+1$, $p(x_{t+1} \mid y_{1:t+1})$.

Next step or terminate: if more observations are available, repeat from the

start of the update loop, otherwise terminate.

For the algorithm thus described an approximation to the observation likelihood can be calculated. The exact likelihood is given by

$$p(y_{1:t}) = p(y_1) \prod_{t=1}^{T+1} p(y_{t+1} \mid y_{1:t}),$$

where

$$p(y_{t+1} \mid y_{1:t}) = \int p(y_{t+1} \mid x_{t+1}) p(x_{t+1} \mid y_{1:t}) dx_{t+1},$$

The distribution $p(x_{t+1} \mid y_{1:t})$ is approximated by by the particle collection after resampling and propagation, but before re-weighting by the observation:

$$\hat{p}(x_{t+1} \mid y_{1:t}) = \frac{1}{N_{t+1}} \sum_{i=1}^{N_{t+1}} \frac{p(x_{t+1}^i \mid x_t^{a_{t+1}^i})}{q(x_{t+1}^i \mid x_t^{a_{t+1}^i}, y_{1:t+1})} \frac{\nu_t^i}{R(a_{t+1}^i \mid \nu_t)} \delta_{\{x_{t+1}^i\}}.$$

The resampling distribution R is effectively just another importance distribution in a similar way as q, which allows some particles to be over- or under-sampled compared to their weight ν_t^i, as noted in [223] as part of the generalized sequential importance sampling framework given there. An appropriate choice of R can be used to give auxiliary particle filters, for example (see section 2.3.2). Given the weights in equation (4.7) this gives

$$\hat{p}(y_{t+1} \mid y_{1:t}) = \frac{1}{N_{t+1}} \sum_{i=1}^{N_{t+1}} w_{t+1}^i,$$

and

$$\hat{p}(y_{1:t+1}) = \prod_{t=1}^{T} \frac{1}{N_t} \sum_{i=1}^{N_t} w_t^i. \tag{4.8}$$

The joint distribution of the particle state and ancestor variables using this algorithm is given by

$$\psi(x_{0:T}, a_{1:T}) = \prod_{i=1}^{N_0} p(x_0^i) \prod_{t=0}^{T-1} \prod_{j=1}^{N_t} R(a_{t+1}^j \mid \nu_t) q(x_{t+1}^j \mid x_t^{a_{t+1}^j}). \tag{4.9}$$

4.3.2 Particle Independence Metropolis-Hastings (PIMH) Sampler

The PIMH sampler applies Metropolis-Hastings sampling to the extended space of all variables ($x_{0:T}$ and $a_{1:T}$) generated in the above particle filter algorithm, along with one additional selector variable k, which serves to select a particular particle from those drawn. The insight behind the PIMH algorithm is that this sampling can be arranged so that the marginal distribution of the selected samples $x_{0:T}^k$ is the required target distribution $p(x_{0:T} \mid y_{1:T})$.

Proposals are drawn by running the particle filter above and then drawing a particle k from the final filter distribution (i.e. with probability ν_T^k). The ancestral path of this particle (notated $x_{0:T}^k$) is then used as a sample for the path $x_{0:T}$, so that the sample is

$$x_{0:T}^k = \left(x_0^{b_0}, x_1^{b_1}, ..., x_{T-1}^{b_{T-1}}, x_T^{b_T} \right),$$

where b_t is the index of the selected particle's ancestor at time t so that $b_{t-1} = a_t^{b_t}$ for $t = 1, ..., T$ with $b_T = k$. The proposal distribution for all variables is therefore

$$Q(x_{0:T}, a_{1:T}, k) = \psi(x_{0:T}, a_{1:T}) \nu_T^k. \tag{4.10}$$

The target distribution over all the variables $x_{0:T}$, $a_{1:T}$ and k can be anything with the target distribution $p(x_{0:T} \mid y_{1:T})$ as the marginal distribution for $x_{0:T}^k$. Therefore, the following target distribution is suitable (noting that

$a_{t+1}^{b_{t+1}} = b_t$).

$$\begin{aligned}\tilde{\pi}(x_{0:T}, a_{1:T}, k) &= \tilde{\pi}(x_{0:T}^k, a_{1:T}^k, k)\tilde{\pi}(x_{0:T}^{-k}, a_{1:T}^{-k} \mid x_{0:T}^k, a_{1:T}^k, k) \\ &= \frac{p(x_{0:T}^k \mid y_{1:T})}{\prod_{t=0}^T N_t} \frac{\psi(x_{0:T}, a_{1:T})}{p(x_0^{b_0}) \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})} \quad (4.11)\end{aligned}$$

The first fraction in equation (4.11) corresponds to the marginal $\tilde{\pi}(x_{0:T}^k, a_{1:T}^k, k) = \tilde{\pi}(a_{1:T}^k, k)\tilde{\pi}(x_{0:T}^k \mid a_{1:T}^k, k)$. The distribution of $x_{0:T}^k$ (conditional on the chosen path) is given by the target distribution, so that $\tilde{\pi}(x_{0:T}^k \mid a_{1:T}^k, k) = p(x_{0:T}^k \mid y_{1:T})$, and the prior on the chosen path $\tilde{\pi}(a_{1:T}^k, k) = \left(\prod_{t=0}^T N_t\right)^{-1}$, corresponding to the $a_{1:T}^k$ and k each being uniformly distributed over the particle collection at the corresponding time given only $x_{1:T}^k$. This is exactly what occurs in a particle filter with resampling, because although a particular $x_{0:T}^k$ is selected according to its weight, its location in the set of all variables is effectively uniform random over all paths owing to the resampling step that randomly selects ancestors for each particle in the new generation.

The second fraction in equation (4.11) is the conditional distribution of the variables $x_{0:T}^{-k}$ and $a_{1:T}^{-k}$, conditioned on the k^{th} path, given by $x_{0:T}^k$, $a_{1:T}^k$ and k. This distribution is given by the conditional distribution of those variables as generated by the particle filter, which can be seen by considering the distribution of the particle filter variables in equation (4.9), conditioned on a particle path $b_{0:T}$, i.e.

$$\begin{aligned}\psi(x_{0:T}^{-b_{0:T}}, a_{1:T}^{-b_{0:T}} \mid x_{0:T}^{b_{0:T}}, a_{1:T}^{b_{0:T}}) &= \prod_{i=1, i\neq b_0}^{N_0} p(x_0^i) \prod_{t=0}^{T-1} \prod_{j=1, j\neq b_t}^{N_t} R(a_{t+1}^j \mid \nu_t) q(x_{t+1}^j \mid x_t^{a_{t+1}^j}) \\ &= \frac{\psi(x_{0:T}, a_{1:T})}{p(x_0^{b_0}) \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})} \quad (4.12)\end{aligned}$$

Since the marginal distribution of $\tilde{\pi}(x_{0:T}, a_{1:T}, k)$ over $x_{0:T}^k$, $a_{1:T}^k$ and k is proportional to the posterior of interest, samples drawn from $\tilde{\pi}(x_{0:T}, a_{1:T}, k)$ will have $x_{0:T}^k$, $a_{1:T}^k$ and k marginally distributed according to $p(x_{0:T}^k \mid y_{1:T})$, as required. In order to target this distribution using proposals from equation (4.10) it is necessary to calculate an acceptance ratio, and for this to be

tractable. The acceptance ratio is given by

$$p_{\text{accept}} = \max\left(1, \frac{\tilde{\pi}(x'_{0:T}, a'_{1:T}, k')}{Q(x'_{0:T}, a'_{1:T}, k')} \frac{Q(x_{0:T}, a_{1:T}, k)}{\tilde{\pi}(x_{0:T}, a_{1:T}, k)}\right),$$

where the $x_{0:T}$ are the current values of the x variables and $x'_{0:T}$ are the proposed values, and similarly for other variables. The key element of this acceptance ratio is the ratio of target and proposal distributions, given by

$$\frac{\tilde{\pi}(x_{0:T}, a_{1:T}, k)}{Q(x_{0:T}, a_{1:T}, k)} = \frac{p(x^k_{0:T} \mid y_{1:T})}{p(x_0^{b_0})\nu_T^k \prod_{t=0}^{T} N_t \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})},$$

which can be seen from equations (4.10) and (4.11). The denominator here can be re-written as

$$p(x_0^{b_0})\nu_T^k \prod_{t=0}^{T} N_t \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t}) = \frac{p(x^k_{0:T}) p(y_{1:T} \mid x^k_{0:T})}{\hat{p}\,(y_{1:T})}. \quad (4.13)$$

The derivation of this key step is given in Appendix C. This means that the ratio

$$\frac{\tilde{\pi}(x_{0:T}, a_{1:T}, k)}{Q(x_{0:T}, a_{1:T}, k)} = \frac{\hat{p}\,(y_{1:T})}{p(y_{1:T})}, \quad (4.14)$$

and the acceptance ratio is given by

$$p_{\text{accept}} = \max\left(1, \frac{\hat{p}'\,(y_{1:T})}{\hat{p}\,(y_{1:T})}\right), \quad (4.15)$$

where $\hat{p}'(y_{1:T})$ is the approximate likelihood obtained from the particle filter used to generate the proposal, and $\hat{p}\,(y_{1:T})$ is the approximate likelihood obtained from the particle filter used to obtain the current sample. Since these are calculated from the particle filter, this acceptance ratio is tractable and thus the method can be used to draw exact samples from the target.

4.3.3 Particle Marginal Metropolis-Hastings (PMMH) Sampler

The method described above only allows samples to be drawn from $p(x_{0:T} \mid y_{1:T})$ as long as the parameters of the system do not change from one sample to the next. However, it is straightforward to extend the method to allow sampling from $p(x_{0:T}, \theta \mid y_{1:T})$, where θ are system parameters (or some subset of them).

The proposal distribution Q in equation (4.10) can be modified to include a proposal from a parameter proposal distribution $q_\theta(\theta' \mid \theta)$, becoming

$$Q(x_{0:T}, a_{0:T-1}, k) = \psi(x_{0:T}, a_{0:T-1}) q_\theta(\theta' \mid \theta) \nu_T^k. \tag{4.16}$$

The target distribution can then be extended to have the true joint distribution of the parameters and states $p(\theta, x_{0:T} \mid y_{1:T})$ as a marginal, with $p(\theta, x_{0:T}^k \mid y_{1:T})$ replacing $p(x_{0:T}^k \mid y_{1:T})$ in equation (4.11). Following the same logic as above, the acceptance ratio for this proposal and target pair is given by

$$p_{\text{accept}} = \max\left(1, \frac{\hat{p}'(y_{1:T} \mid \theta') q_\theta(\theta \mid \theta') p(\theta')}{\hat{p}(y_{1:T} \mid \theta) q_\theta(\theta' \mid \theta) p(\theta)}\right). \tag{4.17}$$

Applying this method to individual parameters or small blocks allows it to be used as part of a Metropolis-within-Gibbs scheme for parameter estimation problems where the likelihood $p(y_{1:T} \mid \theta)$ is not tractable.

4.3.4 Particle Gibbs (PGibbs) Sampler

The Particle Gibbs algorithm is slightly different in flavour to those above; instead of using the output of a particle filter as a proposal that is accepted with some probability, it uses a modified particle filter algorithm, the conditional SMC sampler, to directly sample conditional distributions of the extended target $\tilde{\pi}$. This is particularly useful in cases where an efficient Gibbs sampler can be devised for $p(\theta \mid x_{0:T}, y_{1:T})$, since that can then still

be used for parameter estimation, with the PGibbs method being used to draw samples of the state space variables.

The PGibbs sampler algorithm to sample from $p(\theta, x_{0:T} \mid y_{1:T})$ uses a Gibbs sampler to first sample $p(\theta \mid x_{0:T}, y_{1:T})$ from its full conditional, and then sample $p(x_{0:T} \mid \theta, y_{1:T})$ by sampling from the extended target $\tilde{\pi}$ and taking the marginal $x_{0:T}^k$ as a sample from the required target. The algorithm is

- Sample $\theta \sim p(\theta \mid x_{0:T}, y_{1:T})$.
- Sample $X_{0:T} \sim p(x_{0:T} \mid \theta, y_{1:T})$ via the steps:

  Sample $x_{0:T}^{-k}, a_{1:T}^{-k} \sim \tilde{\pi}(x_{0:T}^{-k}, a_{1:T}^{-k} \mid k, x_{0:T}^k, a_{1:T}^k, \theta)$,

  Sample $k \sim \tilde{\pi}(k \mid x_{0:T}, a_{1:T}, \theta)$,

  $X_{0:T} = x_{0:T}^k$ is a sample from $p(x_{0:T} \mid \theta, y_{1:T})$.

Note that in this sampler the variables $x_{0:T}^{k_{old}}$ and $a_{1:T}^{k_{old}}$, where k_{old} is the incoming sample of k, are not resampled and remain unchanged. If k does not change from one sample to the next, these variables never get resampled and so the algorithm relies on k changing in order to achieve good mixing across all variables.

Sampling from $\tilde{\pi}(x_{0:T}^{-k}, a_{1:T}^{-k} \mid k, x_{0:T}^k, a_{1:T}^k, \theta)$ involves running a particle filter that samples all paths except the one specified by k. This is because this conditional of the target is given by the second fraction in equation (4.11), so that

$$\begin{aligned} \tilde{\pi}(x_{0:T}^{-k}, a_{1:T}^{-k} \mid k, x_{0:T}^k, a_{1:T}^k, \theta) &= \frac{\psi(x_{0:T}, a_{1:T})}{p(x_0^{b_0}) \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})} \\ &= \psi(x_{0:T}^{-b_{0:T}}, a_{1:T}^{-b_{0:T}} \mid x_{0:T}^{b_{0:T}}, a_{1:T}^{b_{0:T}}), \end{aligned} \tag{4.18}$$

by equation (4.12), which is the conditional particle filter distribution for all variables, conditioned on those states and ancestors in the path selected by k. Thus, this distribution can be sampled by running a particle filter, but keeping the states and ancestors of the path selected by k unchanged. This

does not mean that the weights of this path need to remain unchanged, as these are not sampled variables. Therefore, when it comes to selecting a new path, there is some probability that the previous path (or some early part of it) will be re-selected.

In order to select a path, a sample must be drawn for k from

$$\begin{aligned}\tilde{\pi}(k \mid x_{0:T}, a_{1:T}, \theta) &\propto \tilde{\pi}(x_{0:T}, a_{1:T}, k \mid \theta) \\ &= \frac{\hat{p}\,(y_{1:T})}{p(y_{1:T})}\psi(x_{0:T}, a_{0:T-1})\nu_T^k \\ &\propto \nu_T^k. \end{aligned} \tag{4.19}$$

This can be seen from equation (4.14) and the definition of Q in equation (4.10). Since the collection ν_T forms a normalized probability distribution over k, ν_T^i gives the probability of choosing $k = i$ here.

4.3.5 Smoothing in PMCMC Proposals

The PGibbs algorithm above can suffer from the problem of poor mixing owing to the fact that that variables involved in the previously selected path are not resampled. This problem can be overcome somewhat by changing the relationship between k and the b_t variables to be non-deterministic, which allows the introduction of smoothing in the selected paths. This was proposed in two responses [224; 225] to the original PMCMC paper [33], with the former of these giving a description of the scheme.

In the formulation above, b_t is chosen deterministically to give the path of a selected particle k and the particle $k = i$ is chosen with probability ν_T^i. As noted in section 2.3, a path from the particle filter $x_{0:T}^k$ selected according to this probability is an approximate draw from $p(x_{0:T} \mid y_{1:T})$. However, as shown in e.g. [146], the smoothing estimates obtained in this simple way from the particle filter are frequently rather poor, particularly in early stages, owing to the loss of early stage particle diversity due to resampling.

An alternative proposal mechanism is to draw proposals from the smoother

distribution using the forward-filtering backward-sampling method of [146]. This is done by choosing

$$p(b_T = i) = \nu_T^i,$$

and then

$$p(b_t = i \mid b_{t+1}) \propto \nu_t^i p(x_{t+1}^{b_{t+1}} \mid x_t^i), \tag{4.20}$$

which can be shown to draw samples from the approximate smoother distribution. The probability of choosing $b_t = i$ is calculated by calculating the expression on the right of equation (4.20) for each particle i at time t and then normalizing to give $p(b_t = i) = u_t^i$ with

$$u_t^i = \frac{\nu_t^i p(x_{t+1}^{b_{t+1}} \mid x_t^i)}{\sum_{j=1}^{N_t} \nu_t^j p(x_{t+1}^{b_{t+1}} \mid x_t^j)}.$$

Using this method changes the proposal distribution for the PIMH algorithm to

$$Q(x_{0:T}, a_{0:T-1}, b_{0:T}) = \psi(x_{0:T}, a_{0:T-1}) \nu_T^{b_T} \prod_{t=0}^{T-1} u_t^{b_t}, \tag{4.21}$$

where $b_{0:T}$ replace k as the path selection variables. The target distribution $\tilde{\pi}$ can be adjusted straightforwardly to account for this by setting

$$\tilde{\pi}(x_{0:T}, a_{1:T}, b_{0:T}) = \frac{p(x_{0:T}^k \mid y_{1:T}) \prod_{t=0}^{T-1} u_t^{b_t}}{\prod_{t=0}^{T} N_t} \frac{\psi(x_{0:T}, a_{1:T})}{p(x_0^{b_0}) \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})}. \tag{4.22}$$

This target retains $p(x_{0:T} \mid y_{1:T})$ as its marginal with respect to $x_{0:T}^k$ and leaves the key ratio $\tilde{\pi}/Q$ unchanged from that in equation (4.14), leading to a tractable acceptance probability.

For the PGibbs method, the conditional distribution $\tilde{\pi}(b_{0:T} \mid x_{0:T}, a_{1:T}, \theta)$ must be sampled in place of that for k in equation (4.19). From the new target and proposal distributions in equations (4.22) and (4.21), and from

the ratio $\tilde{\pi}/Q$ from equation (4.14), this can be found as

$$\begin{aligned}\tilde{\pi}(b_{0:T} \mid x_{0:T}, a_{1:T}, \theta) &\propto \tilde{\pi}(x_{0:T}, a_{1:T}, b_{0:T} \mid \theta), \\ &= \frac{\hat{p}\,(y_{1:T})}{p(y_{1:T})}\psi(x_{0:T}, a_{0:T-1})\nu_T^k \prod_{t=0}^{T-1} u_t^{b_t}, \\ &\propto \nu_T^k \prod_{t=0}^{T-1} u_t^{b_t},\end{aligned}$$

and so $b_{0:T}$ can be drawn so as to sample from the approximate smoother distribution.

In our work in chapter 6, the use of the smoother in this way has been found to substantially improve early-stage performance in the PGibbs methods, even in light of the additional computational effort; see sections 4.5 and 6.7. Backward sampling requires evaluation of the state transition density, so this method is not applicable in situations in which this is not available. For models where the transition density is intractable, the only available Particle MCMC method is one based around a simulation-only bootstrap particle filter with no backward sampling. For such models, alternative methods such as those based on the construction of bridging distributions as proposed in [33] might be necessary to overcome early-stage path degeneracy in the particle filter.

4.4 Jump Inference within Particle MCMC methods

The Particle MCMC methods described in section 4.3 employ a standard particle filter to evaluate the likelihood approximation used in the acceptance ratio of the PIMH and PMMH algorithms, and a conditional version of the standard filter for sampling in the PGibbs method. The VRPF algorithm described in section 3.3.1 is not a completely standard particle filter of the form used in the Particle MCMC methods of [33], and so the use of such a filter within those methods must be established as valid. There are two possible approaches to this: a proof that the VRPF algorithm as stated in sec-

tion 3.3.1 works within the Particle MCMC algorithms to produce samples from the correct target distribution, or modification of the VRPF algorithm in such a way that it can be cast as a standard particle filter, to which the methods and proofs of [33] apply directly. Here, the second approach is taken, as it can draw on existing work in framing the VRPF as a standard particle filter in [226] and [227].

The filtering problem of estimating jump times and system state can be factorized as

$$p(X_{1:T}, \mathcal{T}_{1:T} \mid y_{1:T}) = p(X_{1:T} \mid \mathcal{T}_{1:T}, y_{1:T}) p(\mathcal{T}_{1:T} \mid y_{1:T}).$$

The conditional distribution of the states given the jumps is linear Gaussian and so can be determined via standard methods (e.g. Kalman filtering), which corresponds to Rao-Blackwellization of the filter. It is therefore sufficient for the particle filter (and therefore the PMCMC methods) to estimate the jump parameters given the observations, by sampling from $p(\mathcal{T}_{1:T} \mid y_{1:T})$.

Following the development in [226] and [227], define a Markov process $(\tau_j, \theta_j)_{j \in \mathbb{N}}$ consisting of jump times $\tau_j \in \mathbb{R}^+$ and their parameters $\theta_j \in \Theta$. In the case of the Gaussian jumps described previously, Θ consists of a finite set of values indicating which element(s) of the state process (e.g. position x or trend $\dot{x}$) the jump took place in. A general choice, however, is $\Theta = \mathcal{X}$, where $\mathcal{X}$ is the state space of the target being tracked, so that $X_t \in \mathcal{X}$ for all t. This allows the jump parameter θ to be specified as a jump size in any (or several) components of the state. Define a continuous time counting process ν_t, counting the number of jumps from time 0 to time T as

$$\nu_t = \sum_{j=1}^{\infty} \mathbb{I}_{[0,t]}(\tau_j),$$

where $\mathbb{I}_{[0,t]}(x)$ is 1 if $0 \leq x \leq t$ and 0 otherwise. Also, let $k_n = \nu_{t_n}$ be the number of jumps to the n^{th} observation time, occurring at t_n. The system

state at a time t_n (i.e. immediately after the n^{th} observation), is given by $Z_n = (k_n, \tau_{1:k_n}, \theta_{1:k_n})$, where

$$Z_n \in E_n = \bigcup_{k=0}^{\infty} \{k\} \times \Upsilon_{n,k} \times \Theta^k$$

with $\Upsilon_{n,k} = \{\tau_{1:k_n} : 0 < \tau_1 < ... < \tau_{k_n} < t_n\} \subset (\mathbb{R}^+)^k$. The E_n form a series of state spaces for the n^{th} state, with $E_{n-1} \subset E_n$ because $\Upsilon_{n-1,k} \subset \Upsilon_{n,k}$. These can all be embedded in a state space E

$$E = \bigcup_{k=0}^{\infty} \{k\} \times \Upsilon_k \times \Theta^k$$

with $\Upsilon_k = \{\tau_{1:k} : 0 < \tau_1 < ... < \tau_k\} \subseteq (\mathbb{R}^+)^k$, and $\Upsilon_{n,k} \subset \Upsilon_k$, so that $E_n \subset E$ for all n. Thus, $Z_n \in E$ for all n.

The state transition density $p(Z_n \mid Z_{n-1})$ for this system is given by

$$\begin{aligned} p(Z_n \mid Z_{n-1}) &= p(k_n, \tau_{1:k_n}, \theta_{1:k_n} \mid k_{n-1}, \tau_{1:k_{n-1}}, \theta_{1:k_{n-1}}) \\ &= S(t_n, \tau_{k_n}) \prod_{j=k_{n-1}+1}^{k_n} p(\tau_j \mid \tau_{j-1}) p(\theta_j \mid \tau_j, \theta_{j-1}, \tau_{j-1}), \quad (4.23) \end{aligned}$$

where $S(t_n, \tau_{k_n}) = p_\tau(\tau > t_n \mid \tau_{k_n})$, a survivor function giving the probability that no jump occurs between τ_{k_n} and t_n. The observation density $p(y_{t_n} \mid Z_{1:n})$ is given by

$$p(y_n \mid Z_{1:n}) = \int p(y_n \mid X_{t_n}) p(X_{t_n} \mid Z_{1:n}) dX_{t_n}.$$

For the conditionally linear Gaussian model $p(X_{t_n} \mid Z_{1:n})$ is given by $\mathcal{N}(X_{t_n}; \mu_{t_n}, \Sigma_{t_n})$, where μ_{t_n} and Σ_{t_n} can be found (as a function of $Z_{1:n}$) using the Kalman filter as described in section 3.2.1, and the initial state prior $p(X_0)$. For linear Gaussian observations $p(y_n \mid Z_{1:n}) = \mathcal{N}(y_n; X_{t_n}, \Sigma_{obs})$, and so, since the observation noise has zero mean and is independent of the state noise,

$$p(y_n \mid Z_{1:n}) = \mathcal{N}(y_n; \mu_{t_n}, \Sigma_{t_n} + \Sigma_{obs}). \quad (4.24)$$

Thus, the trans-dimensional state-space filtering problem can be cast as a standard state space filtering problem with the state at each time in the fixed state space E, and the state sequence after the n^{th} observation $Z_{1:n} \in E^n$.

Mapping from this set-up to the one previously introduced is straightforward, since $\mathcal{T}_{S:T}$ is the set $\{(\tau_i, \theta_i) : S \leq \tau_i \leq T\}$, ordered by time, and k_n is the number of jumps from time 0 to t_n.

Resampling

The VRPF algorithm given in algorithm 3 in chapter 3 uses a residual resampling scheme. This is a valid resampling scheme within the PIMH and PMMH schemes, as noted in [33], as long as a further step is introduced after resampling that assigns each offspring particle to a random index in the next generation. This is necessary because an assumption in the derivation of PMCMC methods is that the probability of a successor of a given index i having a particular ancestor can be calculated and has the same support as the incoming particle collection (i.e. all non-zero weighted particles have a non-zero chance of being selected). Unmodified residual resampling, as used in the VRPF in chapter 3, cycles through each particle in the current generation and chooses a number of offspring, which are assigned into the next available offspring indices. Under this scheme, the calculation of a particular offspring having a given parent is not obvious. Neither does an offspring particle have a non-zero chance of having any non-zero weight parent.

However, as noted in [228], the Particle Gibbs scheme in [33] was only established under the assumption of Multinomial resampling. [228] relaxes this assumption to allow residual resampling (even without the randomization step, provided residual resampling is used at every stage), as well as systematic resampling. Thus residual resampling, as used in the VRPF algorithm of chapter 3, can be used within all types of Particle MCMC methods.

Algorithm

The particle fitler algorithm for jump inference within Particle MCMC methods can be stated as shown in algorithm 5.

Unbiasedness

A key property of particle filters that allows them to be used in PIMH and PMMH algorithms is that they provide an unbiased estimate of the observation likelihood $p(y_{1:T} \mid \theta)$. This means that the marginal of the target distribution is indeed the required target distribution, allowing pseudo-marginal methods such as PIMH and PMMH to be constructed correctly. A proof of the unbiasedness of the particle filter observation likelihood estimate is given in [125], which holds for the (standard) particle filter given in algorithm 5.

Collapsing Offspring

The variable rate filter in chapter 3 used a scheme in which offspring of a given particle having no jumps between the current and next observation times were collapsed into a single particle, which was then assigned increased weight. In order to keep the algorithm as a standard particle filter for use in Particle MCMC methods, this is not done here. However, it is worth keeping track of particles with the same ancestor and no jumps in the next period, since this allows weight computation to be performed only once for this group of particles, since they are identical. This idea can also be extended to multiple stages by keeping track of all groups of identical particles at each stage. All particles in the next generation that are children of one of these particles and have no new jumps added will have identical states, allowing weight computation to be shared.

This calculation sharing is simply a way of increasing computational efficiency and is not necessary.

Algorithm 5 Particle Filter for Jump Inference within PMCMC

Initialization ($n = 0$): Initialize N particles with no jumps, i.e. $Z_0^{(p)} = \{k_0^{(p)}, \tau_{1:k_0}^{(p)}, \theta_{1:k_0}^{(p)}\}$, with $k_0^{(p)} = 0$, $\tau_{1:k_0}^{(p)} = \emptyset$, $\theta_{1:k_0}^{(p)} = \emptyset$ for $p = 1, ..., N$

while *observations available* **do**

$n = n + 1$

Observe y_n (observation at time t_n)

Sample Ancestors: Either multinomial or residual resampling with index randomization

Multinomial: sample $a_n^{(p)} \sim \mathcal{M}(a_n^{(p)}, v_{n-1}^{1:N})$ for all $p = 1, ..., N$, where $\mathcal{M}(\cdot, v^{1:N})$ is the multinomial density with weights $v^1, v^2, ..., v^N$

Residual: sample # of offspring $o_{n-1}^{(p)} = \lfloor N v_{n-1}^{(p)} \rfloor + M^{(p)}$ with $M^{(p)} \sim \mathcal{M}(N - R, \bar{v}_{n-1}^{1:N})$, where $R = \sum_{p=1}^{N} \lfloor N v_{n-1}^{(p)} \rfloor$ and $\bar{v}_{n-1}^{i} = \frac{N v_{n-1}^{(p)} - \lfloor N v_{n-1}^{(p)} \rfloor}{N-R}$ for all $p = 1, ..., N$; Assign ancestors $a_n^{(i)}$ randomly such that $o_{n-1}^{(p)}$ particles at step n have ancestor p

foreach *particle* $p \in 1, ..., N$ **do**

Propose new state $Z_n^{(p)} = \{k_n, \tau_{1:k_n}, \theta_{1:k_n}\}$ from proposal density q:

$$Z_n^{(p)} \sim q(Z_n^{(p)} \mid Z_{n-1}^{a_n^{(p)}}, y_{1:n})$$

Weight: calculate unnormalized weight w_n^p as

$$w_n^p = \frac{p(Z_n^{(p)} | Z_{n-1}^{a_n^{(p)}}) p(y_n | Z_n)}{N q(Z_n^{(p)} | Z_{n-1}^{a_n^{(p)}}, y_{1:n})}$$

where $p(Z_n^{(p)} \mid Z_{n-1}^{(p)})$ and $p(y_n \mid Z_n)$ are given by equations (4.23) and (4.24), respectively

end

Normalize particle weights to give normalized weights $v_t^{(p)}$ such that

$$v_n^{(p)} = \frac{w_n^{(p)}}{\sum_{p=1}^{N} w_n^{(p)}}$$

Result: Particle collection approximates posterior filtering density after n^{th} observation

end

Conditional Particle Filter

For Particle Gibbs methods, a conditional particle filter is used, in which new particle paths are drawn conditional on a single retained particle path. In that case all variables other than those on the selected path are resampled. This is arranged by performing standard resampling and propagation (next jump sampling) steps as described, but generating $N-1$ successor particles at $t+1$ and taking care not to assign the index of the selected (non-sampled) particle at $t+1$ during the resampling step. The selected particle up to $t+1$ is then added to this collection at its specified index. Weighting in light of the observation is carried out as normal.

4.4.1 Sampling Jumps: Particle Gibbs

The Particle Gibbs method described in section 4.3.4 offers an alternative method to reversible jump MCMC for sampling jump times. In this case, the conditional VRPF described in section 4.4 is used to sample the jump times and types $\mathcal{T}$ from $p(\mathcal{T} \mid \theta, y_{1:T})$ (to which backward sampling can be added as described in sections 3.4 and 4.3.5).

4.4.2 Sampling Paramters and Jumps: PMMH

The PMMH algorithm in section 4.3.3 offers a way of using the likelihood estimates obtained from the particle filter to derive acceptance ratios for Metropolis-within-Gibbs sampling of parameters for models in which the likelihood is intractable but for which sample paths can be simulated. A new parameter value for a parameter (or block of parameters) θ_i can be proposed from a proposal distribution and this proposal accepted with probability given by the approximate likelihood ratio from equation (4.17). In this framework, the particle filter can be used to fill the likelihood evaluation role taken on by the Kalman filter when estimating parameters for linear Gaussian models.

Since the PMMH method also produces samples of the state (in this

case the jumps) it is possible to use a collapsed Gibbs sampler similar to that described in section 4.1.2, but with steps 1a and 1b replaced with a single step to sample $\theta_m, \mathcal{T} \sim p(\theta_m, \mathcal{T} |, \theta_X, y)$ via PMMH. In fact, it is most likely that parameters in θ_m will be sampled one at a time from $p(\theta_{m_i}, \mathcal{T} | \theta_{m_{-i}}, \theta_X, y)$, since block proposals are unlikely to be accepted.

4.5 Results

This section presents a series of tests, attempting to compare the various methods for state and parameter estimation proposed in this chapter. A two-factor model similar to that proposed in section 3.5 is used throughout for the evaluation. State estimation methods are compared first using true parameter values, followed by a comparison of parameter estimation methods using true state (jump) values. Better performing methods in each of these areas are then compared when estimating state and parameter values simultaneously. Finally, an attempt is made to estimate parameters of the finance model in section 3.5 on real data. The results of this are compared to the parameters estimated for a similar model in [2].

All implementations of the algorithms used in this section are in Matlab and are not especially optimized. Where possible, components (e.g. the Kalman filter likelihood evaluation) are re-used between algorithms to make comparisons fairer.

4.5.1 State Estimation

There are five methods available for state estimation: PIMH (with and without backward sampling), PGibbs (with and without backward sampling) and RJMCMC. In order to understand the general features of these methods, state estimation was attempted on data generated from the model with known parameters. These true parameters were used during subsequent state estimation. All particle filters were run with a nominal 100 particles (i.e. the number of samples after resampling is 100, though due to the struc-

Parameter	Value
σ_1	0.5
σ_2	0.1
σ_{j_1}	8
σ_{j_2}	3
σ_{obs}	1
θ_1	-0.3
θ_2	-0.2
λ_1	0.001
λ_2	0.1

Table 4.1: Parameter values used in generating results in this chapter for the model described in section 3.5

ture of the VRPF algorithm, some of these are likely to be identical and thus collapsed together).

The parameters used for all the tests in this section are given in table 4.1. Data generated from these parameters is typical of the sort of data generated by this model (other than sequences that grow exponentially). The jump rate λ_1 in the x_1 process is low, so that almost all jumps are found in the x_2 process. An example of observations generated using these parameters can be seen in figure 4.3, in which the observations are shown in green in the top panel; the underlying x_1 process is shown in light grey in that panel, and can also be seen in the top panels of figures 4.5-4.6.

The results in figures 4.3 and 4.5-4.6 show inferred jump positions in the x_1 and x_2 processes in the first two panels, with the grey bars indicating the proportion of samples in which a jump was inferred to be present in each time period between successive observations (i.e. between t and $t+1$, meaning that the grey bars are integer-aligned). The red bars indicate the true positions of jumps, with the intensity of their colour indicating the relative intensity of the jump. A scaled version of the corresponding process (black line) is superimposed in these panels. The final panel shows the total number of accepted proposals against the number of proposals made. For the PGibbs sampler, this is calculated by looking at the number of samples that differ from the previous one.

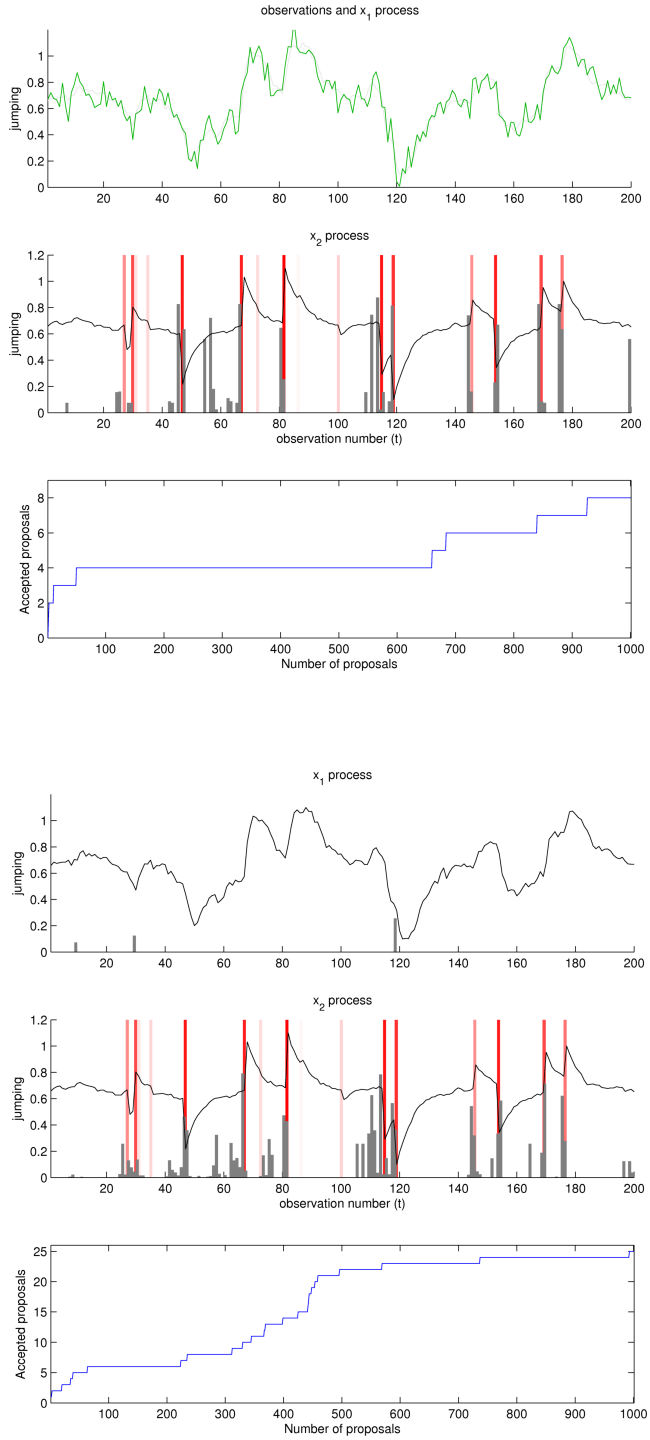


Figure 4.3: PIMH state estimation without (top) and with (bottom) backward sampling. Top panel shows observations in green, x_1 process in light grey. Second panel shows x_2 process in black, true jumps as red bars and proportion of samples containing a jump in each inter-observation time period as grey bars. Third panel shows number of accepted samples against number of proposals. Remaining panels show are as above, for the method with backward sampling

The results for the PIMH sampler in figure 4.3 illustrate the low acceptance rates that can plague this algorithm. Without backward sampling, the acceptance rate is less than 1%. Backward sampling significantly improves the acceptance rate to 2.5% in this test, although this rate is still low. For both of these algorithms, especially that without backward sampling, this leads to very peaky jump-time distributions. However, these methods accurately infer the position of all the major jumps in the sample. The burn-in period for the figures shown is 100 samples. The run time to produce the samples was 6018s and 6463s without and with backward sampling, respectively, meaning that back-sampling adds about 7.4% to the runtime. Figure 4.4 examines the impact of the number of nominal particles on the acceptance rate and runtime for this algorithm with backward sampling. Increasing the number of particles appears to increase the acceptance rate approximately linearly in the range examined, though further tests with very high numbers of particles would be interesting. It is to be expected that at a certain point the return from increasing particle number will diminish. Runtime increases linearly with number of particles, as would be expected.

The results using the PGibbs algorithms in figure 4.5 show good jump estimation, with much better sample diversity (each sample is different from the previous one). There is very little visible difference between the results with and without backward sampling of jump times. This is due to the Rao-Blackwellization of the particle filter, with only jumps being included in the particle state. This means that particle ancestor diversity degenerates more slowly. For long time series it is likely that backward sampling would perform better. The burn-in period for the figures shown is 100 samples. The run time to produce the samples was 6167s and 6519s without and with back sampling, respectively, meaning that back-sampling adds about 6% to the runtime in this case. The backward sampling PGibbs algorithm is about 1% slower than the equivalent PIMH algorithm.

The RJMCMC state sampling approach illustrated in figure 4.6 also pro-

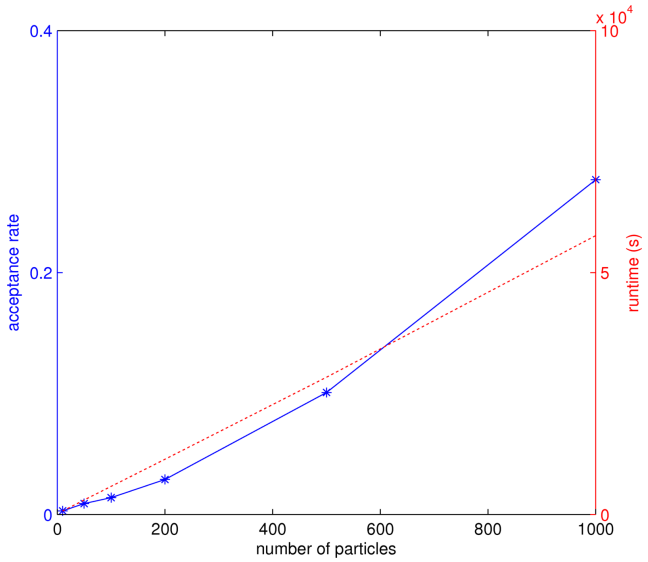


Figure 4.4: Acceptance rate (blue, solid) and runtime (red, dotted) for PIMH algorithm with back-sampling with varying number of (nominal) particles

duces good jump time estimation with high sample diversity (the acceptance rate here is around 40%). More RJMCMC samples have jumps in areas between the true jumps than the PGibbs samples; however, it is not clear whether this is a better or worse representation of the posterior, and in both cases the level of these false positives is low. The burn-in period for the figure 4.6 is 1000 samples, and the runtime to produce the samples was 892s, meaning that RJMCMC is around 70 times faster than the PMCMC methods per sample, though each accepted sample only differs slightly from the previous one.

Though it is not evident here (probably because of the fairly short state sequences involved and the Rao-Blackwellized structure of the filtering problem), results in chapter 6 show that the addition of backward sampling to PGibbs improves sample diversity, especially in early parts of the state sequence. Thus, the conclusion from these results is that PGibbs and RJMCMC produce the best results when attempting to exactly sample the state sequence using known parameter values. PIMH methods suffer form very high rejection rates and so do not produce good sample diversity.

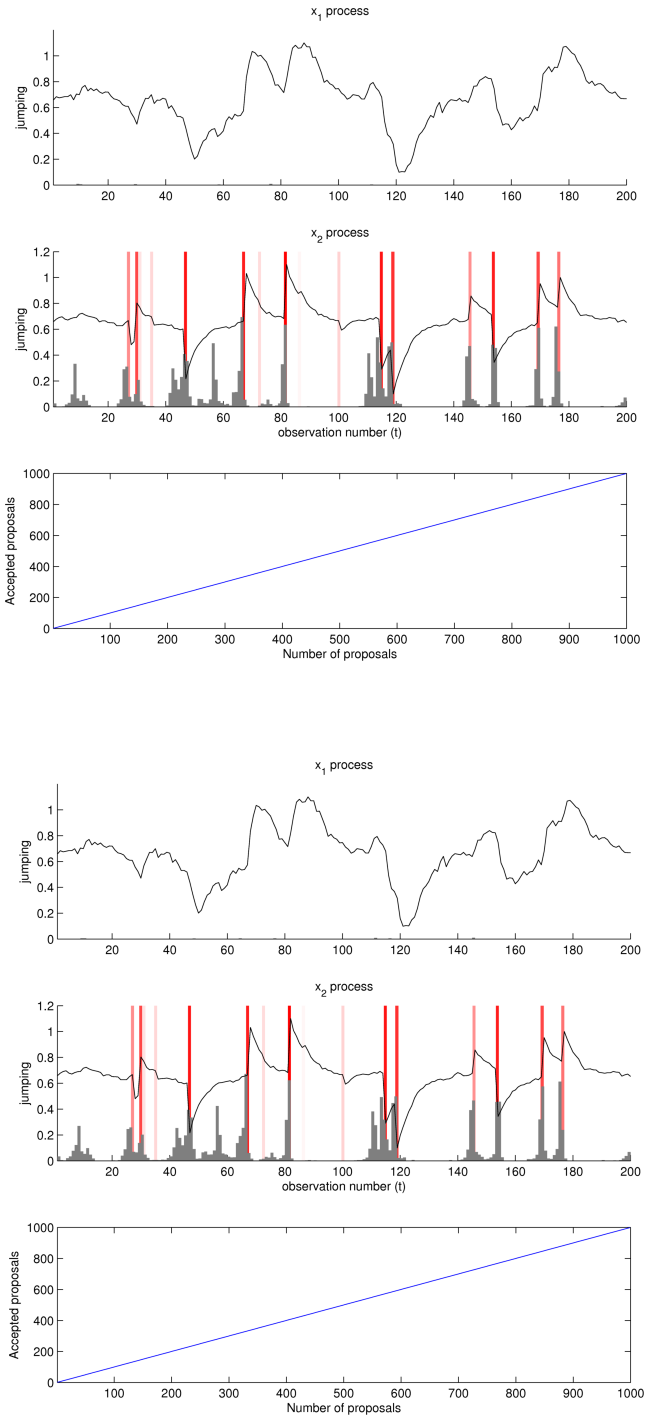


Figure 4.5: PGibbs state estimation without (upper three panels) and with (lower three panels) backward sampling

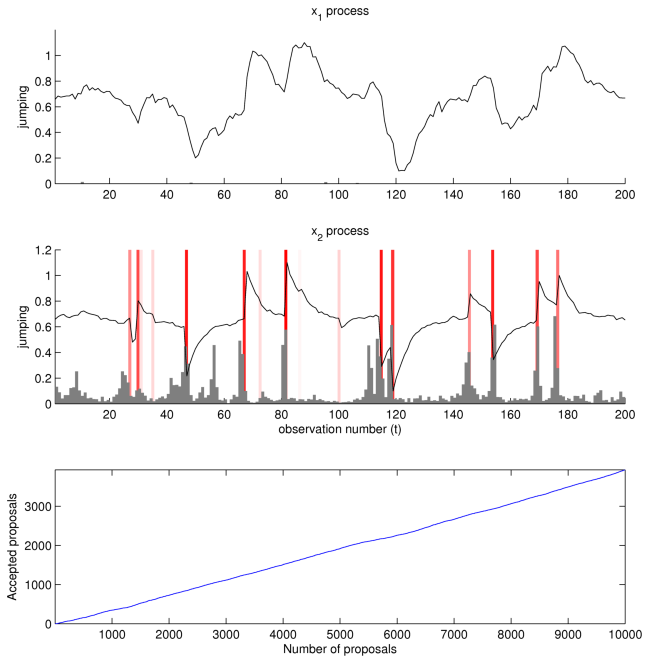


Figure 4.6: RJMCMC state estimation

4.5.2 Parameter Estimation

Three different parameter estimation methods (conditional on the jump sequence) have been proposed: marginalized Gibbs sampling in which the state sequence X is not sampled; sampling the state X, and then using Gibbs sampling on this sample to estimate parameters; and a collapsed Gibbs sampler that selectively uses the previous two methods for different parameters. This section compares these methods by applying them to parameter estimation problems with known jumps. For the model considered here it is always possible to estimate jump rates directly from a jump sequence and so these are Gibbs-sampled directly in all cases.

Inverse gamma priors were used for the observation noise variance σ^2_{obs} and for the jump variance $\sigma^2_{j_{\{1,2\}}}$. This distribution was chosen because it provides a conjugate prior when estimating these values directly from samples of either states X or jump times; it is also fairly heavy tailed (see figure 4.7), allowing for the possibility of very large jump scales. An al-

Parameter	Prior	Parameters
Diffusion variance $\sigma^2_{\{1,2\}}$	Gamma	$\alpha = 1, \beta = 2$
Jump size variance $\sigma^2_{j_{\{1,2\}}}$	Inverse Gamma	$\alpha = 1, \beta = 1$
Jump size std. dvn. $\sigma_{j_{\{1,2\}}}$	Uniform	$a = 3\sigma^2_i, b = 10\sigma^2_i$
Observation variance σ^2_{obs}	Inverse Gamma	$\alpha = 1, \beta = 1$
Negated mean reversion $-\theta_{\{1,2\}}$	Gamma	$\alpha = 1, \beta = 10$
Jump rate $\lambda_{\{1,2\}}$	Gamma	$\alpha = 3, \beta = 1$

Table 4.2: Distributions and hyper-parameters for the parameter priors used in parameter estimation

ternative prior for the jump variance is a uniform distribution of between a and b times the size of the diffusion noise variance for the process. This type of prior is perhaps a better encoding of actual prior expectations of jump scale, which can be expected to significantly exceed the scale of the diffusion process (since otherwise a 'jump' is fairly meaningless). As it destroys conjugacy, this type of prior has only been used with the marginalized sampler as it is straightforward to incorporate in the Metropolis-within-Gibbs sampler used for jump size variance there. Gamma priors were used for the diffusion noise variances $\sigma^2_{\{1,2\}}$, the (negated) mean reversion coefficients $-\theta_{\{1,2\}}$, and jump rates $\lambda_{\{1,2\}}$. The gamma distribution is a conjugate prior for the jump rates and was chosen as a prior for the diffusion noise and mean reversion coefficients because it offered flexible, vague priors that could be biased to the areas in which these parameters are realistically expected to be (fairly near to zero for the mean reversion rates, with very little chance of being smaller than -1, and more likely to be smaller than larger for the diffusion noise variance). These prior beliefs are reflected in the (hyper-)parameters chosen for the prior distribution, given in table 4.2, with the resulting priors shown in figure 4.7. In reality, however, for the large number of observations in a series (typically several hundred) these priors have a very limited effect compared to the data (except in regions where they are zero).

Figures 4.8-4.11 show the results of parameter estimation for a sequence of observations generated from the model using the range of parameter

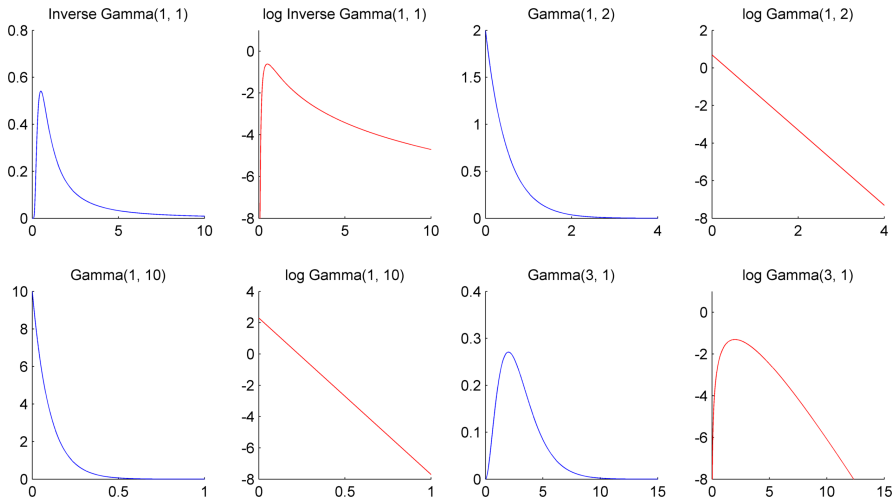


Figure 4.7: Priors (blue) and their logs (red) for the parameters given in table 4.2

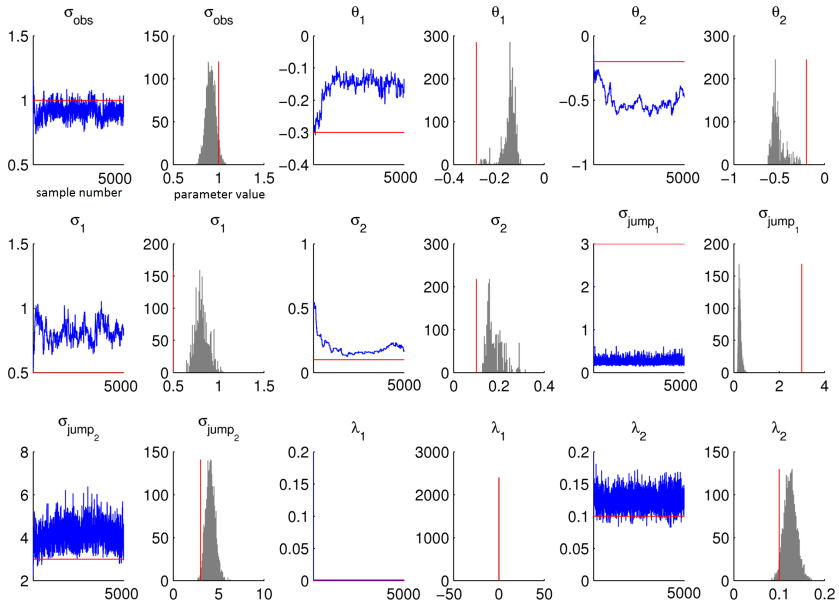


Figure 4.8: Parameter estimation by including X in state space (true parameter values indicated by red lines). For each parameter, left chart (blue horizontal line) shows state sequence of MCMC chain, right chart show histogram of MCMC samples

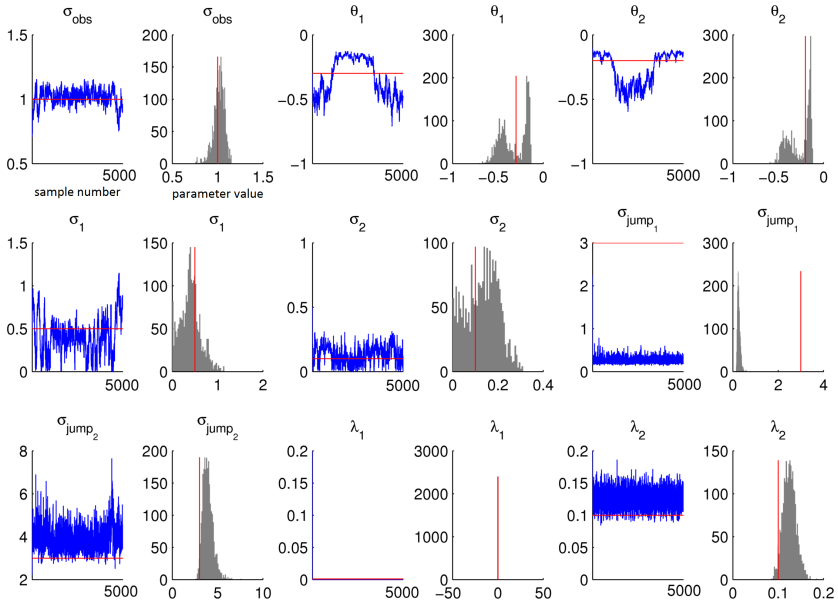


Figure 4.9: Parameter estimation via collapsed Gibbs sampler, using sample of X for estimation of jump variance ($\sigma^2_{j\{1,2\}}$)

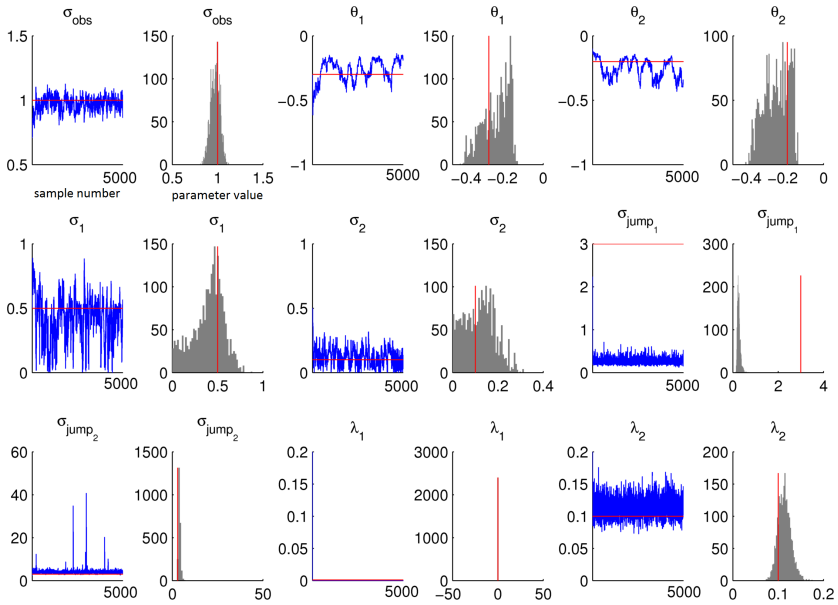


Figure 4.10: Parameter estimation using fully marginalized Gibbs sampler

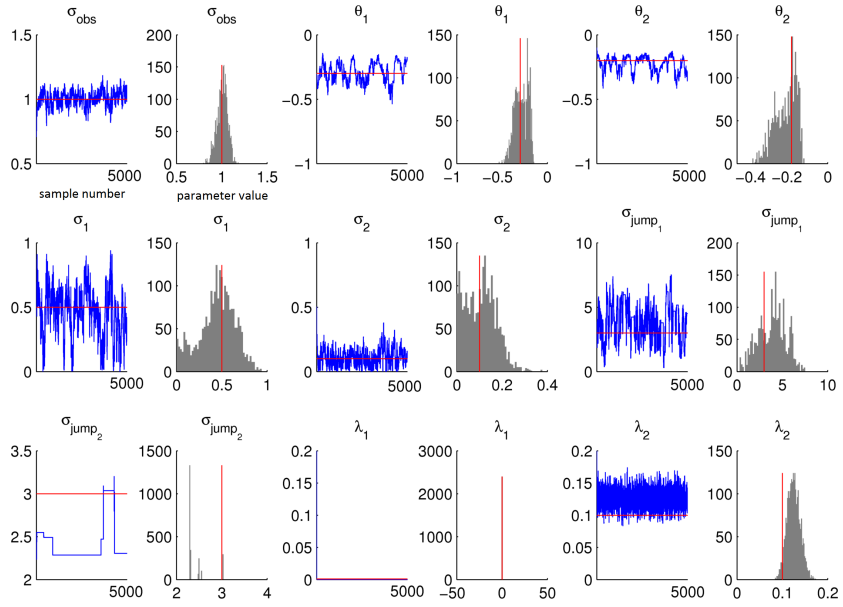


Figure 4.11: Parameter estimation using fully marginalized Gibbs sampler with uniform prior on jump standard deviation $p(\sigma_{j_i} \mid \sigma_i) \sim \mathcal{U}(3\sigma_i, 10\sigma_i)$

estimation algorithms. For each parameter two plots are shown, the first (blue line) showing the parameter value evolution with sample number (useful for gauging convergence) and the second (grey bars) showing a histogram of sampled parameter values post-burn in. The true parameter value is marked with a red line in both cases. The estimates in figures 4.8-4.11 were generated using a sequence of 600 observations.

Figure 4.8 shows the parameter estimation results using a sampled state sequence X. The estimates are reasonably close to the true values in most cases, although estimation of the mean reversion coefficients θ_1 and θ_2 looks to converge to incorrect values. The shape of the likelihood with respect to these variables makes this sort of mis-estimation somewhat likely (see section 4.5.3). It is not clear that the estimate for σ_2 has converged at all. Because there were no jumps in the x_1 process, there was no data available from which to estimate σ_{j_1}; it was therefore sampled from its prior, leading to poor estimation.

The collapsed sampler used in figure 4.9 (in which jump variances are

estimated from a sample of the state X) achieves good parameter estimation, with the true parameters falling within the estimated distributions for all parameters (other than σ_{j_1}). The sample of the state sequence X could also have been used to estimate the observation noise variance σ^2_{obs}, however this leads to failure of the algorithm. Inclusion of σ^2_{obs} in the parameters estimated by using a sample of X appears to lead to instability, with σ^2_{obs} increasing uncontrollably and leading all other parameters to become unstable until numerical failure occurs. This is probably because increasing σ^2_{obs} means that observations have increasingly little influence on the sampled state sequence, leading to increasingly erratic state sequences being sampled. This causes parameter estimates derived from these state sequences to begin to diverge, resulting in a feedback loop that produces ever more extreme state sequence samples (especially with respect to jump magnitude), rapidly leading to numerical instability and failure of the estimation. Jump rates are estimated directly from the jump sequence and all other parameters are estimated using the marginalized Gibbs sampler.

The marginalized samplers used in figures 4.10 and 4.11 seem to perform very well, with rapidly converged estimates of the parameters that encompass the true parameter values. In figure 4.11 a uniform prior on the jump variance is used (as opposed to the inverse gamma prior used in the other parameter inference tests in this section). This prior limits the jump variance to lie between 9 and 100 times the diffusion noise variance of the given process, which is designed to encode the belief that jump variance will be significantly larger than the diffusion noise variance, but not without limit. In this test this resulted in very low acceptance rates for proposals for the σ_{j_2} parameter, albeit in a range close to the correct value.

In all these tests, the jump rate λ_2 was slightly over-estimated. Given this is directly inferred from the true jump sequence in these tests, this bias is perhaps surprising. The most likely explanation is that it is caused by the inverse Gamma prior applied to the jump rate (with parameters $\alpha = 1, \beta = 1$), which has a maximum at 0.5.

4.5.3 Exploration of Likelihood

Estimation of certain parameter values has proved difficult in this model. Of particular interest are two pairs of parameters that seem to interact strongly with each other to cause difficulty and ambiguity in their estimation, namely the observation noise variance σ^2_{obs} and x_1 process diffusion noise volatility σ_1; and the two mean reversion parameters λ_1 and λ_2. In both cases, fairly strong negative correlation has been observed between the two parameters during some estimation attempts. Figure 4.12 shows the shape of the log-likelihood function for a certain set of 800 generated observation from the model for both parameter pairs in the region of the true parameter values. This reveals that both pairs of variables are negatively correlated in regions of high likelihood (dark red). The mean reversion coefficients in particular show a very flat likelihood surface around a large range of values in which the parameter values can be reversed without a very large effect on the log-likelihood. This goes some way to explaining the switches seen between parameter values in figure 4.9. Figure 4.13 shows the log-likelihood function with respect to the mean reversion parameters for a selection of randomly selected parameter values. These show that the strength of the L-shape in the log-likelihood varies significantly with respect to the mean reversion and other parameters, with mean reversion parameters near 0 showing the strongest such effects. Estimation of these parameters is likely to be more difficult and ambiguous if they fall in those ranges.

In general, the likelihood function for this parameter estimation problem is fairly flat around the true parameters. This can be seen by looking at the likelihood function evaluated using the true jumps at a number of points in parameter space with parameters perturbed by up to some amount from the true parameters. Table 4.3 shows the results of this for parameters perturbed by up to 10%, 25%, 50% and 100% from their true values (by adding uniform random values to each true parameter scaled to the appropriate percentage of the true value) and for random parameter

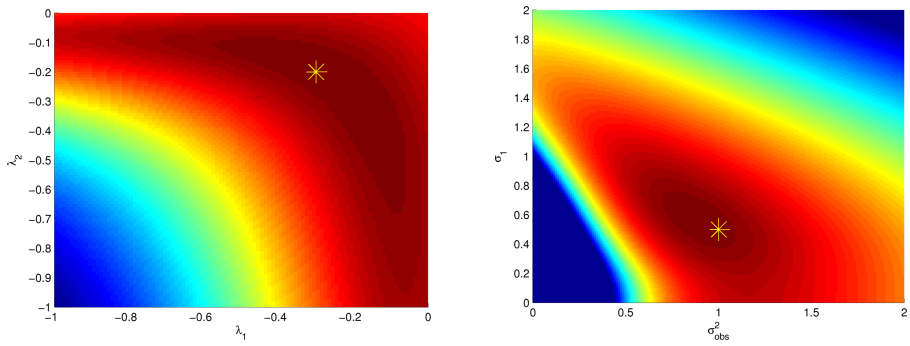


Figure 4.12: Log-likelihood function with respect to mean reversion parameters (left) and σ^2_{obs} and σ_1 (right). All other parameters and jump times set to true values. Yellow star indicates true values

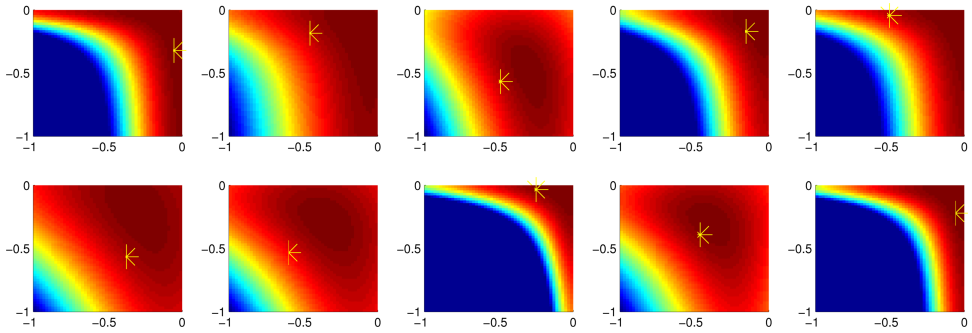


Figure 4.13: Log-likelihood function with respect to mean reversion parameters (λ_1 on horizontal axis, λ_2 on vertical axis) for a selection of randomly chosen parameter values. Yellow star indicates true values

Parameters	Log-likelihood
true	–1055.5
true ±10%	–1056.2 (1.8)
true ±25%	–1065.0 (9.5)
true ±50%	–1093 (35)
true ±100%	–1319 (454)
random	–1371 (410)

Table 4.3: Mean log-likelihood for various parameter sets (standard deviation shown in brackets where applicable) using true jump positions

values within reasonable ranges. The log-likelihood values are were calculated on data generated from the model consisting of 600 observations; the same data was used for all tests.

The values obtained for the parameters perturbed by 10% and 25% are very close to those for true parameters (and within one standard deviation). Even with 50% perturbation the log-likelihood is only slightly increased. The log-likelihood for random parameters is much greater, indicating that the parameter values do significantly affect the likelihood, but that in the region of the true parameters, the likelihood function is relatively flat. It is therefore likely that parameter estimations will be somewhat diffuse and, especially in cases where jumps are also estimated, could differ significantly from the true parameters while still being plausible parameter values for the data observed.

4.5.4 Parameter and State Estimation

Due to their apparent superiority in state estimation, only the PGibbs with backward sampling and RJMCMC methods were used in the tests in this section. Parameter estimation via collapsed and marginalized Gibbs samplers were tested for parameter estimation, since both these methods performed well in the parameter estimation tests. This gave four possible methods for joint parameter and state estimation, all of which were tested on synthetic data generated using the same parameters as in the parameter estimation tests above. 600 observations were generated and the same data was used

	estimated jumps	true jumps	no jumps
Estimated parameters			
PGibbs, Marginalized	-1078.9 (12.4)	-1079.5	-1272.4
RJMCMC, Marginalized	-1084.1 (18.4)	-1068.9	-1325.0
PGibbs, Collapsed	-1094.3 (11.5)	-1127.1	-1236.7
RJMCMC, Collapsed	-1097.9 (11.3)	-1116.6	-1235.4
RJMCMC (5k), Marginalized	-1116.0 (18.6)	-1079.8	-1204.4
RJMCMC (5k), Collapsed	-1101.7 (12.9)	-1118.1	-1219.1
True parameters			
PGibbs, Marginalized	-1222.7 (55.7)	-1055.5	-2271.8
RJMCMC, Marginalized	-1206.9 (92.1)	-1055.5	-2271.8
PGibbs, Collapsed	-1309.4 (61.3)	-1055.5	-2271.8
RJMCMC, Collapsed	-1309.8 (35.6)	-1055.5	-2271.8

Table 4.4: Mean log-likelihood (standard deviation shown in brackets where applicable) for parameter and jump estimates derived using various state and parameter estimation techniques. RJMCMC (5k) results refer to tests run using 5000 samples

for all tests. For RJMCMC methods, 10000 state and parameter samples were generated, with state and parameters being sampled alternately. A burn-in of 1000 samples was used. These methods took around 6 hours to run. For the PGibbs methods, 800 state samples were generated using a particle filter with 100 particles, with 7 samples of the parameters generated for each state sample, giving 5600 parameter samples. The runtime for these methods was also around 6 hours, with the number of samples being chosen so that the computational effort for both RJMCMC and PGibbs methods was comparable. Figures 4.14-4.17 show the state and parameter estimates obtained from each of these algorithms, and table 4.4 gives the log-likelihood values for parameter and state estimates obtained using each method. In calculating these values, the posterior mean parameters were used as the estimated parameters and a selection of 50 jump time samples chosen randomly from the post-burn in period were used as the estimated jumps.

With PGibbs state estimation, both the collapsed and marginalized sampler

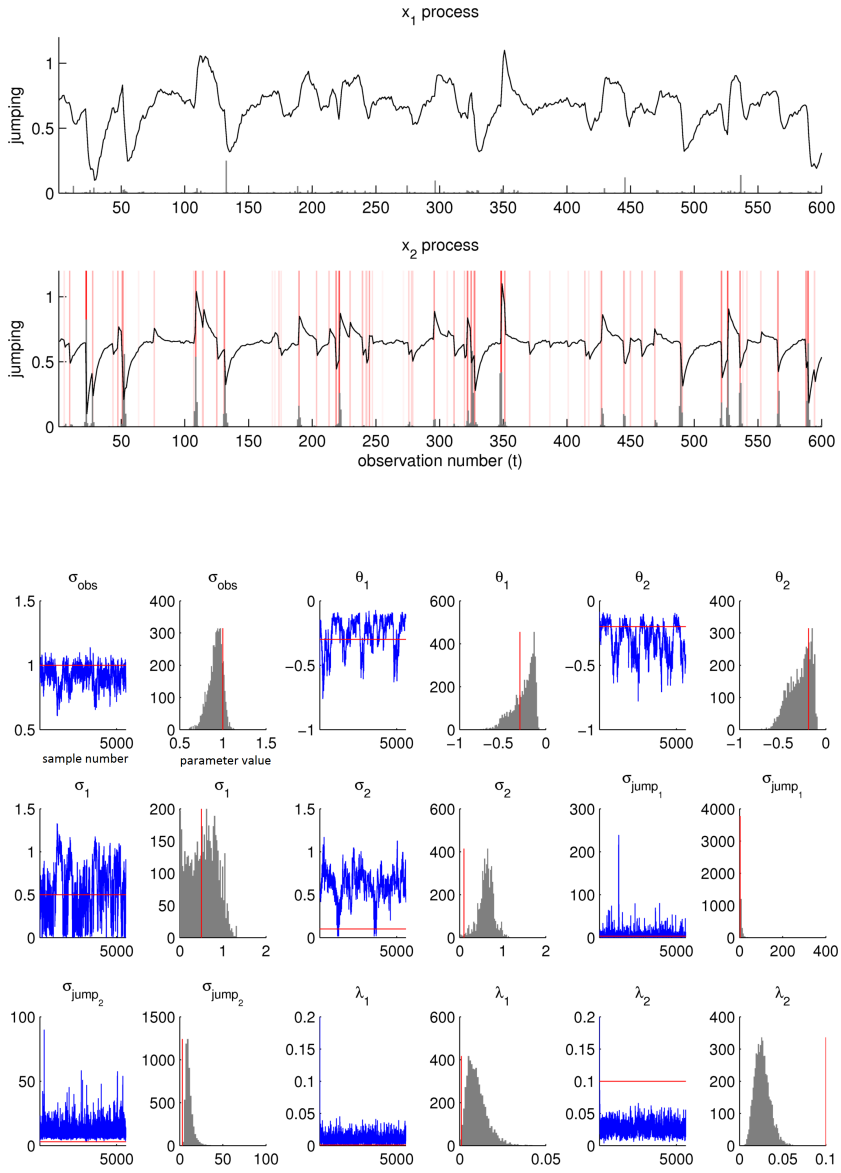


Figure 4.14: State and parameter estimates with PGibbs state estimation, collapsed Gibbs parameter estimation

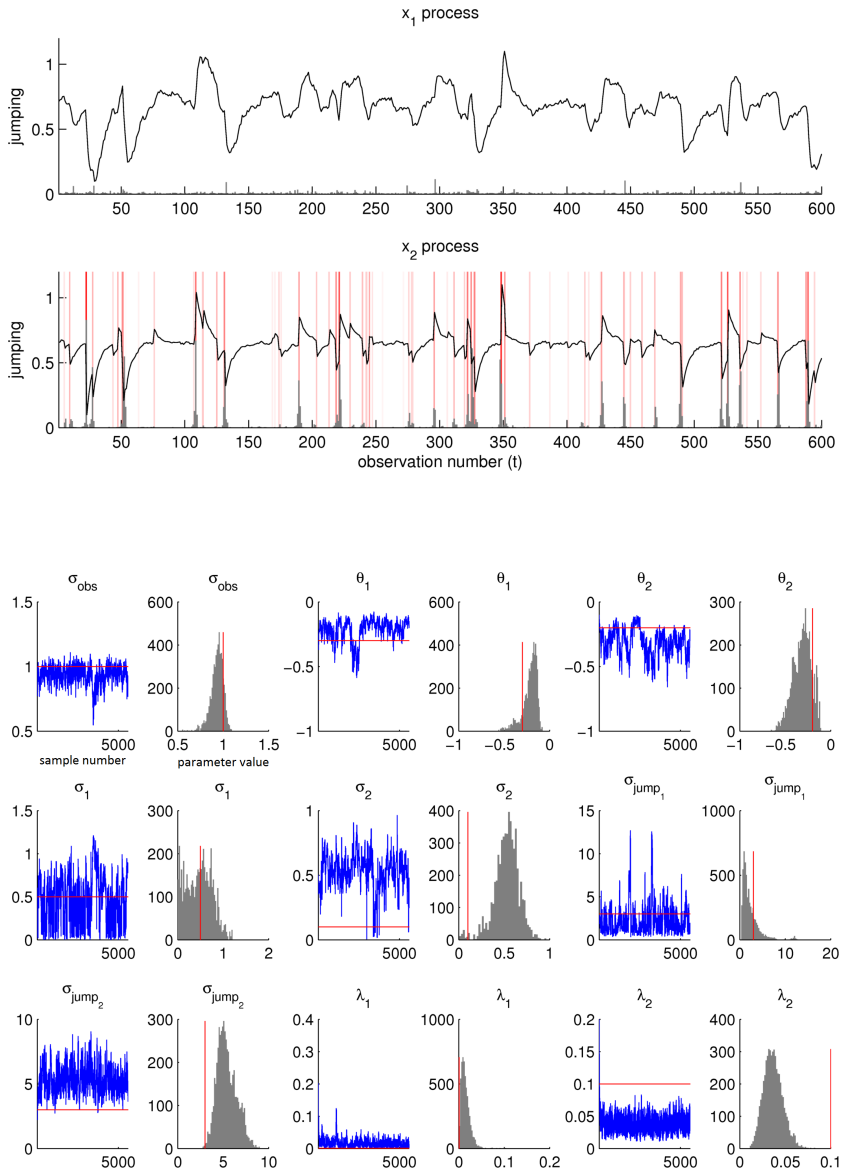


Figure 4.15: State and parameter estimates with PGibbs state estimation, marginalized Gibbs parameter estimation

give good parameter estimation results (figures 4.14 and 4.15). These results and those in table 4.4 show the two methods performing fairly similarly, with the marginalized Gibbs sampler having a slight advantage and giving slightly tighter posterior distributions for the parameters. The jump inference results are good, with the PGibbs method identifying all the major jumps in the sequence without introducing any significant spurious jumps. The algorithm correctly identified that all jumps were in the x_2 process.

With RJMCMC state estimation (figures 4.16 and 4.17) the state estimation results are slightly worse, though still good. Most of the major jumps are identified although in some cases a jump is identified in the x_1 process; in the cases where this happens, consideration of the underlying process suggests that this is usually reasonable, as the x_1 process experienced rapid changes at those points. The RJMCMC method seems more prone to identifying spurious jumps, albeit with low probability. With RJMCMC state estimation the marginalized sampler for parameter estimation again seems to produce slightly better results for parameter estimation, though seems to result in more spurious jumps being identified (with low probability).

For the data in these tests it would appear that PGibbs state estimation is slightly superior to RJMCMC, because it identifies the jumps in the correct processes and produces fewer spurious jumps. For parameter estimation on this data, the fully marginalized Gibbs sampler produces better results than the collapsed sampler, giving better likelihood values in table 4.4 and appearing to give somewhat tighter parameter distributions, with more convincing convergence.

The parameter estimation results in all tests show typical estimation results for this problem, with jump frequency being under-estimated whilst jump scale is over-estimated. This probably reflects the difficulty in distinguishing large diffusion moves from small jumps, so that only larger, more clearly recognizable jumps are classified as such. Examination of the likelihood in table 4.4 suggests that the likelihood function is fairly flat in

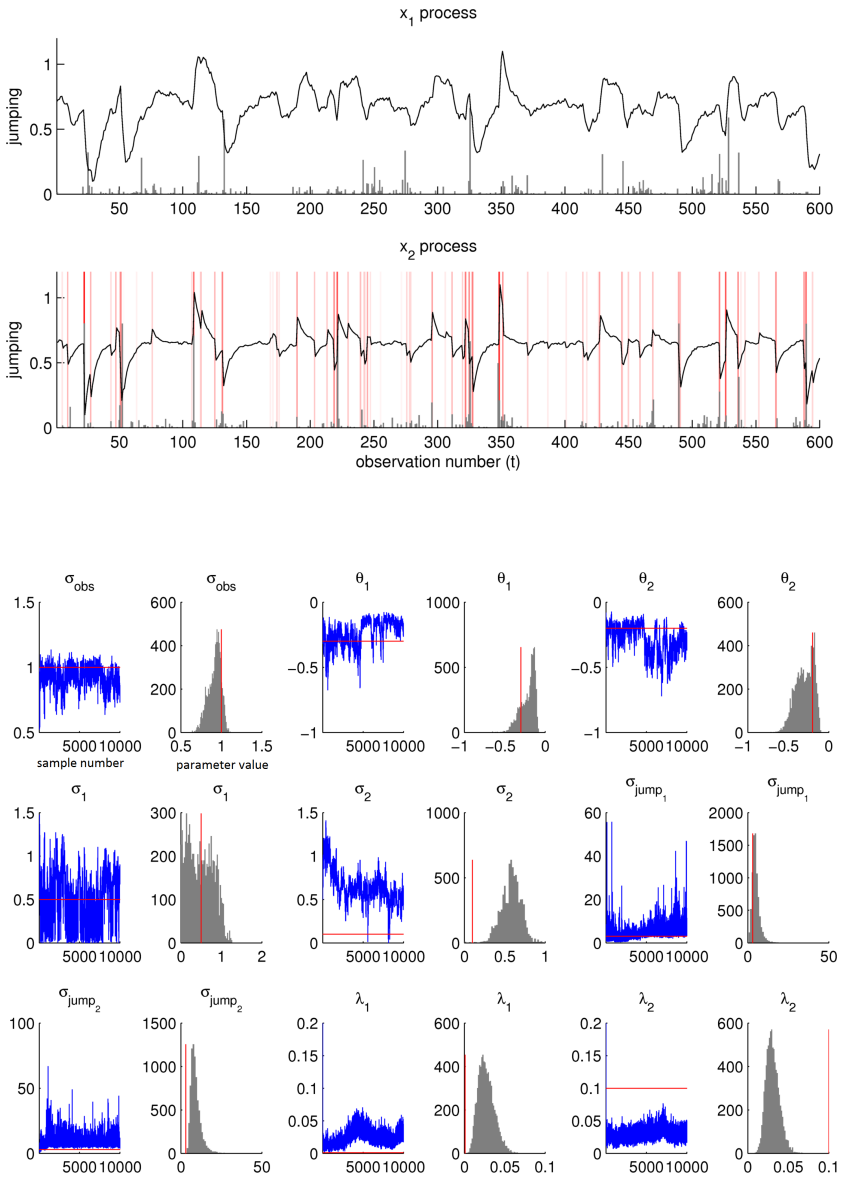


Figure 4.16: State and parameter estimates with RJMCMC state estimation, collapsed Gibbs parameter estimation

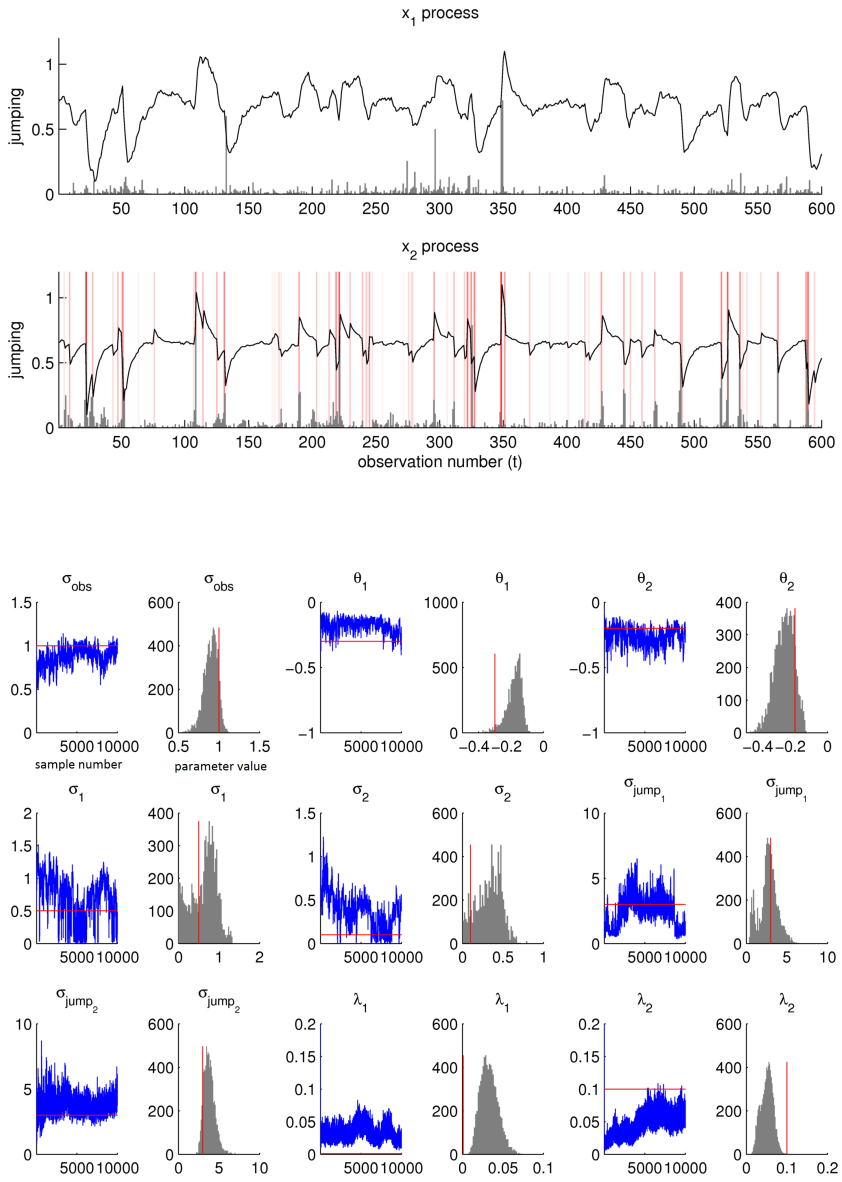


Figure 4.17: State and parameter estimates with RJMCMC state estimation, marginalized Gibbs parameter estimation

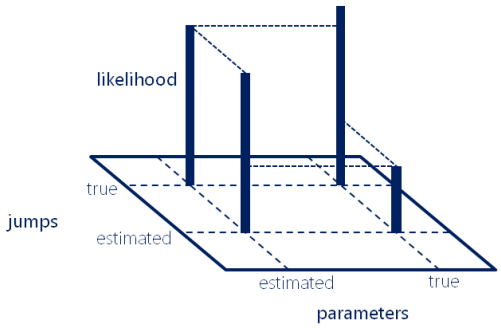


Figure 4.18: Representation of the likelihood values seen in table 4.4

the region of the estimated and true jumps with the estimated parameters, although with the true parameters, the difference between the estimated and true jumps is more marked, as illustrated in figure 4.18. The estimates found seem to at a high-likelihood point in state and parameter space.

Further testing comparing the results of many runs would be desirable to make a full assessment, but the runtime of these tests is around 6 hours for each method, making large scale testing computationally demanding.

Another method, PMMH sampling to implement Metropolis-within-Gibbs for each parameter, is also possible, but the very low acceptance rates of the PIMH algorithm found during state-only estimation (at least with relatively small numbers of particles) led to the dismissal of this idea, since it is likely that the proposal rejection rate will be very high. This method might be plausible if the PIMH algorithm were run with a large number of particles, but the computational burden of this would be very high, although much of the processing for each particle is independent and thus a high number of particles might be achievable via parallelization.

4.5.5 Financial Data

To test parameter estimation on real data, the parameter estimation algorithm was applied to daily data coming from S&P500 index over the period October 2010 to March 2013, comprising about 600 observations. By

Parameter	Estimated value	Value in [2]
σ_{obs}	7.1 (0.46)	20
θ_2	-0.73 (0.14)	-0.2
σ_2	8.4 (1.1)	4.1
σ_{j_2}	45 (7.7)	70
λ_2	0.053 (0.02)	0.2

Table 4.5: Mean parameter estimates (standard deviation) for the Langevin model used in [2], using 600 daily observation of S&P500 from October 2010 to March 2013

using a fairly standard financial series and the Langevin dynamical model from [2] (detailed in section 3.5) it is possible to compare the parameters estimates to those in [2], which were estimated by choosing parameters that produced good portfolio returns in backtesting, rather than directly from the data. Figure 4.19 shows the results of the parameter estimation using this data and model, and table 4.5 gives the parameter estimates. These show that the estimates in [2], though not catastrophically wrong, are some way away from those derived using a principled estimation procedure. The parameters in used [2] overestimated the jump size and frequency (though, as seen earlier, the parameter estimation here is prone to underestimation of jump frequency), and underestimated the diffusion variance. The persistence of trends in the data was also overestimated, with the estimated mean reversion θ_2 being substantially quicker than that estimated in [2]. The rapid speed of the estimated mean reversion indicates that in this data long term trends might not be very prevalent.

Applying the parameter estimation using the full (nine parameter) model on the same data produces the results seen in figure 4.20 and table 4.6. Because the model allows random innovations in the x_1 process, the estimated observation variance is much smaller. The scale of the trend process is greatly reduced (close to zero), again indicating limited long-term trending behaviour (its mean reversion speed is much slower, but this effect is likely to be negated by its small scale).

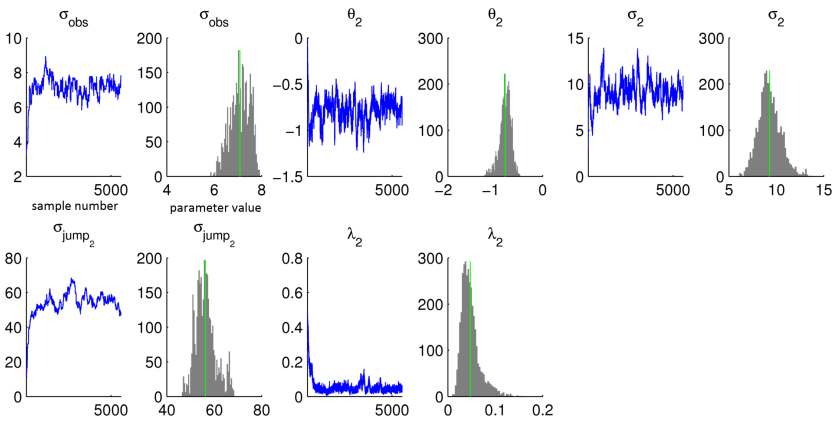


Figure 4.19: Parameter estimation for the Langevin model used in [2], using 600 daily observation of S&P500 from October 2010 to March 2013; green lines show mean parameter value over post-burn in samples

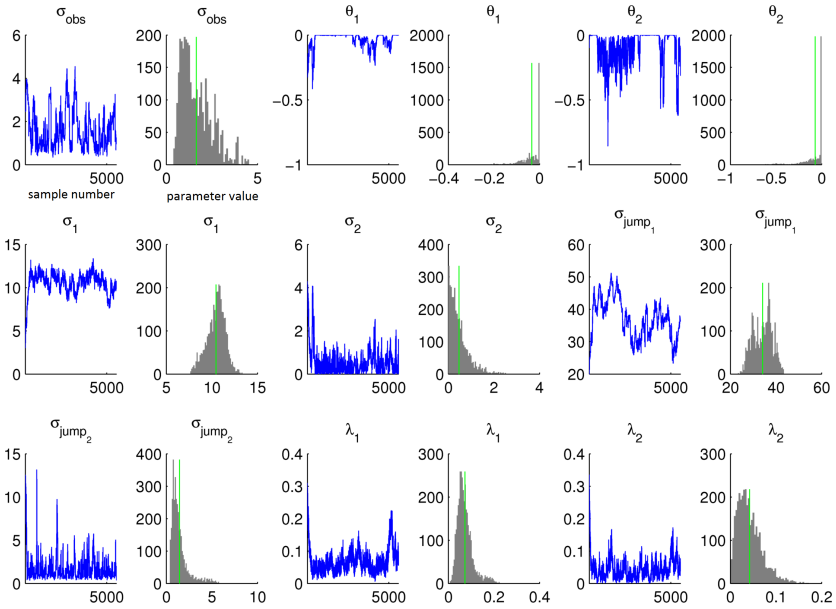


Figure 4.20: Parameter estimation for the full model from section 3.5, using 600 daily observation of S&P500 from October 2010 to March 2013; green lines show mean parameter value over post-burn in samples

Parameter	Estimated value
σ_{obs}	1.6 (0.85)
θ_1	-0.034 (0.043)
θ_2	-0.071 (0.12)
σ_1	10 (1.0)
σ_2	0.47 (0.42)
σ_{j_1}	34 (4.2)
σ_{j_2}	1.4 (0.97)
λ_1	0.073 (0.035)
λ_2	0.042 (0.027)

Table 4.6: Mean parameter estimates (standard deviation) for the full model from section 3.5, using 600 daily observation of S&P500 from October 2010 to March 2013

4.6 Conclusions

This chapter has introduced methods for parameter estimation for linear jump-diffusion models of the type used in [2; 172; 35] and chapter 3. The methods developed are applicable to a wider class of models, including ones with more state processes, non-linear observation models and non-linear state transition models (although those that allow for Rao-Blackwellization have a significant computational advantage). Section 4.4 showed that a modified version of the VRPF used in chapter 3 for jump estimation, can, by following the development of [226], be cast as a standard particle filter, which is compatible with the assumptions of Particle MCMC methods. This allows Particle MCMC methods to be used to estimate the (trans-dimensional) jump sequences encountered in the jump-diffusion models examined here.

It is encouraging that several different estimation methods produce consistent parameter and state estimates. Having a range of methods available allows for at least some degree of verification between them. Though producing similar results, particle MCMC algorithms compared favourably to RJMCMC algorithms for parameter estimation in this problem, and have a number of advantages. PMCMC algorithms are undoubtedly slower than RJMCMC methods for the same number of samples but do not require the

design of proposals, which means that they can be simpler to implement and less sensitive to the proposal mechanism (though in this work very simple proposals were used for the RJMCMC). In order to achieve similar quality results (in terms of the likelihood of the estimated values) it seemed that a similar amount of computational time was required for both PMCMC and RJMCMC methods; using a similar number of samples (as in the RJMCMC (5k) tests in table 4.4) produced somewhat worse results. PMCMC methods can also make use of particle filter methods developed for online state estimation. PMCMC methods lend themselves to parallelization, unlike standard MCMC methods, because although particle filters are interacting systems (via resampling), the evaluation of weights and proposal generation is independent for each particle in a particular generation and thus can be parallelized. This suggests that parallelization of the order of the number of particles can be exploited and as suggested by figure 4.6 hundreds or even thousands of processors (as found on modern GPUs) could yield useful performance improvements.

Tests on real financial data broadly support the choice of parameters in [2], although the specific parameter values are somewhat different (functionally they are likely to be similar). However, parameter estimation on S&P500 data appeared to indicate limited evidence for the presence of trends, at least in the period tested.

Chapter 5

Simultaneous Mapping and Tracking in Potential Field Environments

This chapter presents a new method of simultaneous localization and mapping (SLAM) for objects moving in a potential field environment. Only weak nonparametric assumptions are made about the shape of the potential function through the imposition of a Gaussian process prior. An efficient Bayesian method for the inference of object position and environment structure is presented, based on a Rao-Blackwellized particle filtering scheme. The method improves tracking performance compared to standard tracking methods and reveals hidden structure (such as obstructions) in structured environments, as illustrated by its application to urban car tracking. Applications of the technique demonstrated here include path planning and multi-target tracking applications, in which it is possible to observe (perhaps noisily) some targets moving through the environment. By using this method to learn about environment structure from the passage through it of some targets, better track inference for subsequent targets or better path planning should be possible, by taking this environmental structure into account.

Methods for SLAM usually focus on the problem from the perspective of a robot or other sensor platform receiving local information about its environment from on-board sensors. Here, the slightly different but closely related problem of tracking targets in a structured but unknown environment is addressed. Knowledge of the environment structure can enhance tracking performance by, for example, allowing candidate positions in implausible areas to be rejected. If the target moves in a compact area, or if further targets will be tracked through the area, learning the environment structure can be beneficial to future tracking performance.

The approach taken here attempts to address this problem by modelling the environment structure as a static potential field through which the target moves. This allows the environment structure to influence the motion of the target, with the target likely to move from areas with high potential to those with low potential. For example, a road along which a car can move might be modelled as a low-potential channel, with deviations from this channel resulting in a restoring force back towards its centre. Such environment maps can be inferred from observations of the target as it moves throughout the environment using a formulation similar to that found in traditional SLAM problems.

Prior knowledge of the environment structure and information gleaned from the tracking of other entities in the same environment can also be incorporated in order to create collaborative maps. By putting a non-parametric Gaussian process prior (see e.g. [229]) on the shape of the potential field map, it is possible to infer very general shapes of environment structure, requiring the specification of only a characteristic length scale for the field as a hyperparameter. This scale is often easy to choose, for example the characteristic length scale for modelling roads might be approximately their width, but can also itself be estimated. Since the model is formulated probabilistically and solved in a fully Bayesian way, estimates of the uncertainty in the environment map are also available throughout the domain of interest.

The use of a potential field map, in common with many SLAM setups, leads to a nonlinear state-space formulation of the SLAM problem. In common with other SLAM models, it is possible to derive a Rao-Blackwellized particle filtering method for efficient inference. The results in this chapter are produced using a simulation-based bootstrap particle filter, although it is shown how an adapted particle filter such as that used in [230] could also be applied. The incorporation of a potential field in the motion model leads to a Langevin stochastic differential equation that cannot be solved in closed form. Therefore numerical schemes such as those described in appendix D must be used for both simulation and evaluation of the state transition function. The structure of the environment is inferred via its influence on the motion of the target using a method related to that used in [3], [4] and in chapter 6.

This chapter is organized as follows. Section 5.1 gives a brief overview of related work. Section 5.2 outlines the motion, potential field and observation models necessary for inference. Section 5.3 shows how these models can be used to track a target and infer the potential field in which it moves. Section 5.4 gives illustrative results comparing the proposed model with a standard tracking model using the bootstrap filter for single target tracking with repeated motion through the same environment; and Section 5.5 draws conclusions and suggests further work. Much of the work in this chapter first appeared as my earlier work [3].

5.1 Related Work

Tracking objects in structured environments, where their motion is somehow restricted, is a common problem. For example, cars generally move along roads. If the environment structure is known, e.g. via a map, this information can be used to improve tracking accuracy, as in [231; 232; 233; 166; 234], amongst others, and recently covered in overview in [235]. These take a range of approaches to tracking on roads, including constructing

likelihood functions from the map [235], constraining targets to lie only on roads [234; 232; 233; 166], and using variable interacting models corresponding to different road segments [231]. All these methods rely on *a priori* knowledge of the environment.

In robotics applications, potential fields have been used extensively to constrain motion paths in path planning applications for known environments due to their analytical tractability and flexibility [236]. In such applications targets are modelled as attractive potentials and obstacles modelled as repulsive potentials, encouraging the motion of the robot towards its target whilst avoiding collisions. They were first introduced in robotics in the mid 1980s in [237], and have been studied widely since, with attention paid to the particular form of the potential functions used [238; 239], methods of path planning within them and, more recently, the inclusion of time-varying environment maps [240; 236]. According to the recent review in [241] potential field approaches account for about 11% of path planning algorithms found in the literature in recent years, a figure that has remained steady since their introduction in the 1980s, suggesting they remain applicable.

In many problems the environment structure is *not* known in advance, so hard constraints arising from a map cannot be enforced. In these cases, maps must be learnt in parallel with target tracking. A similar situation is found in SLAM problems where a map of the environment has to be constructed in parallel with sensor localization, with position and map estimates being co-dependent. For example, if a sensor gives range and bearing measurements to some unknown landmarks, then knowledge of the landmark positions allows the sensor position to be inferred, but sensor position is necessary in order to determine the location of the landmarks.

Due to its importance in robot navigation, the SLAM problem has been extensively studied; see the reviews [242; 243]. Of particular interest to the problem considered in this chapter is the formulation of SLAM as a nonlinear state-space problem [244], with the observer location and environ-

ment map jointly forming the system state. Early work used the extended Kalman filter to solve this problem [244] and other nonlinear state inference algorithms such as the unscented Kalman filter have also been applied. More recently, Rao-Blackwellized particle filtering has been applied successfully to the problem, popularized with the introduction of the *FastSLAM* algorithms of [245] and [230].

Though commonly applied to maps of landmarks in space arising from repeatedly encountered visual features, other map structures are also possible. In [246] a grid-based occupancy map of the environment is learnt, again using a Bayesian approach based on the particle filter, with efficiency refinements proposed for such models in [247]. In [248] a map based on finely distributed points is proposed for use with laser rangefinders, in which map points represent object detections by the rangefinder. Such point-cloud maps have also been used with Microsoft's Kinect sensor, a widely available consumer-grade depth camera [249]. Somewhat similar approaches can be used with vision based systems, when the map consists of easily identified visual features such as corners [250]. A different type of map is used in [251], where probabilistic topological maps consisting of graphs of landmarks are constructed, with edges representing adjacency.

In [252] a problem conceptually similar to the one in this chapter is examined. There, the problem is that of learning a map of transportation routes and nodes along with target positions and other information from GPS tracking data. In that work, a complex hierarchical Bayesian model is developed allowing for the use (and inference) of different modes of transport and even the intention of the target. Transport nodes and target goals are determined in an offline EM pass, however, so the algorithm might be unsuitable for sequential learning.

Previous work to estimate road map information from large-scale databases of GPS traces (reviewed, for example in [253]) is also somewhat related, in that it attempts to infer environment structure, in this case road network structure, from tracking data. However, in this case tracking is

not performed simultaneously and the primary aim is to derive road map information, rather than to enhance tracking. The methods employed rely on having a large corpus of GPS traces available for the area of interest.

5.2 Model

The state space model used here consists of three components: a model for the motion of target objects, a model for the potential fields within which they move and a model for the observations made of their position. Specifying these model components allows inference of both object position (tracking) and the potential field (mapping) from noisy observations.

5.2.1 Motion Model

In this work the motion model for tracked objects resembles a near constant velocity model (see e.g. [204]), applied to objects in a potential field. For an object moving in a potential field U, the force exerted by the field on the object is given by the negative gradient of the field at its location $\mathbf{x}_t$, so that $\mathbf{F}_{\text{field}} = -\nabla U(\mathbf{x}_t)$. The object is also assumed to be subject to random forces (corresponding to noisy motion, resistance or internal thrust). This, combined with Newton's second law of motion, gives a second order Langevin SDE for the object's position, which can be written as a pair of first order SDEs as

$$d\mathbf{x}_t = \dot{\mathbf{x}}dt, \qquad d\dot{\mathbf{x}}_t = -\frac{\nabla U(\mathbf{x}_t)}{m}dt + B\,dW_t \tag{5.1}$$

where $\dot{\mathbf{x}}_t$ represents the target's velocity at t, B is the Cholesky decomposition of the noise covariance Σ and dW_t is a vector of the infinitesimal increments of a Gaussian noise process of the same dimensionality as $\mathbf{x}_t$. If independent noise of constant variance σ^2 in each dimension is assumed, $B = \frac{\sigma}{\sqrt{m}}I$. Such SDEs cannot be solved analytically for general fields U (although they can be solved if U is constant, linear or parabolic, in which case they become, in the first two cases the near constant velocity model and in the final case solvable linear SDEs; see chapter 3). In the absence of an ana-

lytic solution, the SDE (5.1) must be numerically integrated, which can be done using, for example, the method in appendix D using the integrator given in equations (D.12) and (D.13). In this case, the $f(\mathbf{x}_t)$ function found in that integrator is given by $-\frac{1}{m}\nabla U(\mathbf{x}_t)$ and the Jacobian of f, J_t, is given by the Hessian of $-\frac{1}{m}U$. The integration method in appendix D allows 'pseudo-observations' of the potential's gradient to be made, as shown in section 5.3.1. These, along with the Gaussian process prior on the form of the potential field (described in the following section), allow the shape of the potential field to be inferred.

5.2.2 Potential Field Prior Model

In order to infer the potential field some prior assumptions about its functional form are required, since otherwise observations made at one point could not be related to its value elsewhere. However, since it is assumed that no prior knowledge of the environment is available, it is desirable to make minimal assumptions about the shape of the field. In order to do this a Gaussian process prior is applied to its functional form. This is a non-parametric prior assumption that allows the shape of the field to take a wide range of forms. A book-length treatment of Gaussian processes is given in [229].

Gaussian Processes

The Gaussian process prior assumption on the shape of a function U can be stated as being the assumption that, at any finite set of points P in the domain of the function, the joint distribution of the corresponding function values $U(p)$ for $p \in P$ is multivariate Gaussian, with mean and covariance being given as deterministic functions μ and K of the point locations, which

can be stated mathematically as

$$p\left(\begin{bmatrix} U(p_1) \\ \vdots \\ U(p_n) \end{bmatrix}\right) \sim \mathcal{N}\left(\begin{bmatrix} \mu(p_1) \\ \vdots \\ \mu(p_n) \end{bmatrix}, \begin{bmatrix} K(p_1,p_1) & \dots & K(p_1,p_n) \\ \vdots & \ddots & \vdots \\ K(p_n,p_1) & \dots & K(p_n,p_n) \end{bmatrix}\right). \quad (5.2)$$

A common choice of μ in the absence of other information is for it to be zero everywhere.

This probabilistic formulation allows the distribution of the function value to be evaluated throughout its domain. For example, if the function value is known at a set of points P, so that $Y = [U(p_1) \dots U(p_n)]'$ is a vector of the values of $U(p)$ for $p \in P$, then the joint distribution of these known values with the values of the function U at a set of 'test points' P_* (where it is unknown) is given by

$$\begin{bmatrix} Y \\ U_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_* \\ K_*' & K_{**} \end{bmatrix}\right), \quad (5.3)$$

where $U_* = [U(p_{*1}) \dots U(p_{*m})]'$ for $p_{*i} \in P_*$ and where the mean has been assumed to be 0 everywhere. Here, the $(i,j)^{\text{th}}$ element of the K matrix is given by the covariance function $K(p_i, p_j)$, and similarly, that of the K_* and K_{**} matrices is given by $K(p_i, p_{*j})$ and $K(p_{*i}, p_{*j})$, respectively. Therefore conditional distribution of U_*, given the known values of U in Y is given by

$$p(U_* \mid Y) \sim \mathcal{N}\left(K_*'K^{-1}Y, K_{**} - K_*'K^{-1}K_*\right). \quad (5.4)$$

Observations distorted by additive Gaussian noise can be incorporated by adding a noise term to the variance of the observations in question. If Z is a set of noisy observations of the function U or its partial derivatives, so that $Z = Y + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \Sigma_{\text{noise}})$, the joint distribution of these noisy

observations with U at a set of test points (as in equation (5.3)) is given by

$$\begin{bmatrix} Z \\ U_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \Sigma_{\text{noise}} & K_* \\ K_*' & K_{**} \end{bmatrix}\right), \tag{5.5}$$

and the conditional distribution of U_* in this case is given by

$$p(U_* \mid Z) \sim \mathcal{N}\left(K_*'(K + \Sigma_{\text{noise}})^{-1}Z, K_{**} - K_*'(K + \Sigma_{\text{noise}})^{-1}K_*\right), \tag{5.6}$$

where K, K_* and K_{**} have the same meaning as before. Other types of observation noise are more difficult to incorporate and require a Gaussian approximation [229]. This allows the distribution of U to be found at an arbitrary set of test points given noisy observations of the function value at various points.

As will be shown in section 5.3.1, the motion of the target can be used to derive a set of noisy observations of the *gradient* of the potential rather than its value. This requires different covariance values to be used describing the covariance between function derivatives and values. These are given in the following section and allow such observations to be easily incorporated.

Covariance Functions and Derivative Observations

In this work, the covariance between the value of the function U at two points $\mathbf{x}_1$ and $\mathbf{x}_2$ is taken to be given by the commonly used *squared exponential* covariance function, so that

$$\text{cov}\left(U(\mathbf{x}_1), U(\mathbf{x}_2)\right) = \exp\left(-\tfrac{1}{2l^2}\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2\right), \tag{5.7}$$

where l is a characteristic length parameter for the Gaussian process U, and can be thought of as a hyperparameter of the process. By choosing a covariance function that results in close correlation of nearby points, local smoothness is favoured (in a probabilistic sense) without making it an absolute requirement. The influence of observations diminishes rapidly bey-

ond a certain distance, allowing wide variation of the function value throughout its domain. Many other covariance functions are possible (see, for example [229]), allowing different situations to be modelled, such as stronger or weaker spatial dependence or periodicity, but these are not considered here.

As is shown below, it is the gradient of the potential U rather than its value that can be (indirectly) observed via its effect on the object. Second partial derivatives of the potential are also needed for the numerical integration scheme in equation (D.12) (since the f that appears there is proportional to the gradient of the potential field ∇U used here). These higher derivatives can be calculated at specific points in a similar way to the function values discussed above. This is done by replacing the appropriate elements of the mean and covariance of the joint distribution in equation (5.8) with values appropriate for calculation of the derivatives.

For the mean, this means that if the i^{th} element of the observation vector (on the left hand side of equation (5.8)) corresponds to a derivative observation, then the i^{th} element of the mean vector of the normal distribution (on the right hand side of equation (5.8)) should be replaced with the corresponding derivative of the function mean. In the case here where the mean is zero everywhere, this too is always zero.

For the covariance, the $(i,j)^{\text{th}}$ element of the covariance matrix must correspond to the covariance between a derivative and whatever type of quantity is in the j^{th} position in the vector on the left. So, for example if both the i^{th} and j^{th} elements are derivatives then the $(i,j)^{\text{th}}$ covariance element must be the covariance between two (partial) derivatives. For example, the joint distribution of the function value at $\mathbf{x}_a$ and its partial derivative with respect to direction $\mathbf{x}_j$ at a point $\mathbf{x}_b$ is given by

$$\begin{bmatrix} U_{\mathbf{x}_a} \\ \frac{\partial U_{\mathbf{x}_b}}{\partial \mathbf{x}_j} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \operatorname{cov}(U_{\mathbf{x}_a}, U_{\mathbf{x}_a}) & \operatorname{cov}\left(U_{\mathbf{x}_a}, \frac{\partial U_{\mathbf{x}_b}}{\partial \mathbf{x}_j}\right) \\ \operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_b}}{\partial \mathbf{x}_j}, U_{\mathbf{x}_a}\right) & \operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_b}}{\partial \mathbf{x}_j}, \frac{\partial U_{\mathbf{x}_b}}{\partial \mathbf{x}_j}\right) \end{bmatrix}\right). \tag{5.8}$$

Therefore, expressions for the covariance between values, first derivatives and second derivatives of U, and all combinations thereof, are required. These can be found by differentiating the covariance between function values [254], since

$$\operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_1}}{\partial \mathbf{x}_j}, U_{\mathbf{x}_2}\right) = \frac{\partial}{\partial \mathbf{x}_j^{(1)}} \operatorname{cov}(U_{\mathbf{x}_1}, U_{\mathbf{x}_2}),$$

where $U_{\mathbf{x}_1}$ is the value of the process U at point $\mathbf{x}_1$ and $\mathbf{x}_j$ is the j^{th} dimension of the domain of U. Here the derivative with respect to $\mathbf{x}_j^{(1)}$ refers to differentiation with respect to the j^{th} dimension of the *first* argument of the covariance function. This idea extends to higher derivatives and to the covariance between derivatives.

For the squared exponential covariance function in equation (5.7) the presentation is simplified by writing $\Delta \mathbf{x}_j = \mathbf{x}_{1,j} - \mathbf{x}_{2,j}$ for the *signed* distance between points $\mathbf{x}_1$ and $\mathbf{x}_2$ in the j dimension and noting that

$$\frac{\partial}{\partial \mathbf{x}_j^{(i)}} \operatorname{cov}(U_{\mathbf{x}_1}, U_{\mathbf{x}_2}) = \frac{\partial(\Delta \mathbf{x}_j)}{\partial \mathbf{x}_j^{(i)}} \frac{\partial}{\partial(\Delta \mathbf{x}_j)} \operatorname{cov}(U_{\mathbf{x}_1}, U_{\mathbf{x}_2}),$$

with

$$\frac{\partial(\Delta \mathbf{x}_j)}{\partial \mathbf{x}_j^{(i)}} = \begin{cases} 1 & i = 1 \\ -1 & i = 2 \end{cases}$$

due to the signed nature of Δx_j. This means that differentiation in the second argument of the covariance leads to negation of the expression. Letting

$$E = \exp\left(-\frac{1}{2l^2} \sum_k (\Delta \mathbf{x}_k)^2\right),$$

the relevant covariance relations are given by

$$\begin{aligned}
\operatorname{cov}(U_{\mathbf{x}_1}, U_{\mathbf{x}_2}) &= \alpha E, && (5.9)\\
\operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_1}}{\partial \mathbf{x}_i}, U_{\mathbf{x}_2}\right) &= -\frac{\alpha}{l^2}\Delta \mathbf{x}_i E && (5.10)\\
\operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_1}}{\partial \mathbf{x}_i}, \frac{\partial U_{\mathbf{x}_2}}{\partial \mathbf{x}_j}\right) &= \frac{\alpha}{l^2}\left(\delta_{ij} - \frac{\Delta \mathbf{x}_i \Delta \mathbf{x}_j}{l^2}\right) E, && (5.11)\\
\operatorname{cov}\left(\frac{\partial^2 U_{\mathbf{x}_1}}{\partial \mathbf{x}_i \partial \mathbf{x}_j}, U_{\mathbf{x}_2}\right) &= \frac{\alpha}{l^2}\left(\frac{\Delta \mathbf{x}_i \Delta \mathbf{x}_j}{l^2} - \delta_{ij}\right) E, && (5.12)\\
\operatorname{cov}\left(\frac{\partial^2 U_{\mathbf{x}_1}}{\partial \mathbf{x}_i \partial \mathbf{x}_j}, \frac{\partial U_{\mathbf{x}_2}}{\partial \mathbf{x}_m}\right) &= \frac{\alpha}{l^4}\left(\frac{\Delta \mathbf{x}_i \Delta \mathbf{x}_j \Delta \mathbf{x}_m}{l^2} - \delta_{ij}\Delta \mathbf{x}_m - \delta_{im}\Delta \mathbf{x}_j - \delta_{jm}\Delta \mathbf{x}_i\right) E, && (5.13)
\end{aligned}$$

and

$$\begin{aligned}
&\operatorname{cov}\left(\frac{\partial^2 U_{\mathbf{x}_1}}{\partial \mathbf{x}_i \partial \mathbf{x}_j}, \frac{\partial^2 U_{\mathbf{x}_2}}{\partial \mathbf{x}_m \partial \mathbf{x}_n}\right) = \\
&\quad \frac{\alpha}{l^6}\Bigg(\frac{\Delta \mathbf{x}_i \Delta \mathbf{x}_j \Delta \mathbf{x}_m \Delta x_n}{l^2} - \delta_{ij}\Delta \mathbf{x}_m \Delta \mathbf{x}_n - \delta_{im}\Delta \mathbf{x}_j \Delta \mathbf{x}_n \\
&\qquad - \delta_{in}\Delta \mathbf{x}_j \Delta \mathbf{x}_m - \delta_{jm}\Delta \mathbf{x}_i \Delta \mathbf{x}_n - \delta_{jn}\Delta \mathbf{x}_i \Delta \mathbf{x}_m \\
&\qquad - \delta_{mn}\Delta \mathbf{x}_i \Delta \mathbf{x}_j + l^2\left(\delta_{in}\delta_{jm} + \delta_{jn}\delta_{im} + \delta_{mn}\delta_{ij}\right)\Bigg) E.
\end{aligned} \tag{5.14}$$

The first two of these are given in [254].

Because $\Delta \mathbf{x}_j$ is signed, swapping the argument in which differentiation occurs leads to the sign of the covariance function being reversed. So, for example,

$$\operatorname{cov}\left(U_{\mathbf{x}_1}, \frac{\partial U_{\mathbf{x}_2}}{\partial \mathbf{x}_j},\right) = -\operatorname{cov}\left(\frac{\partial U_{\mathbf{x}_1}}{\partial \mathbf{x}_j}, U_{\mathbf{x}_2}\right),$$

and similarly for the other expressions.

If noisy observations (with additive Gaussian noise) of some of these quantities are available (first derivatives in the potential field case), they can be incorporated by adding a noise term when considering the variance of that particular observation. For example, noisy first derivative observations would require a noise term to be added to the covariance in equation

(5.11) so that it becomes

$$\text{cov}\left(\frac{\partial U_{\mathbf{x}_1}}{\partial \mathbf{x}_i}, \frac{\partial U_{\mathbf{x}_2}}{\partial \mathbf{x}_j}\right) = \frac{\alpha}{l^2}\left(\delta_{ij} - \frac{\Delta \mathbf{x}_i \Delta \mathbf{x}_j}{l^2}\right) E + \delta_{\{\mathbf{x}_1=\mathbf{x}_2, i=j\}}\sigma^2_{i,\mathbf{x}_1}$$

where $\sigma^2_{i,\mathbf{x}_1}$ is the noise variance of the observation of the first derivative in the i^{th} dimension at $\mathbf{x}_1 (= \mathbf{x}_2$ when applicable).

5.2.3 Observation Model

The inference framework described in Section 5.3 allows for arbitrarily complicated observation functions, subject to the limitations of the particle filter [255]. For the examples in this chapter, however, a simple Gaussian observation model is used, where the observation y_i at time t_i, is the true object position at that time $\mathbf{x}_{t_i}$ distorted by additive Gaussian noise:

$$y_i = \mathbf{x}_{t_i} + \eta_i, \tag{5.15}$$

with $\eta_i \sim \mathcal{N}(0, \sigma^2_{\text{obs}} I)$.

5.3 Inference

The target's state $X = [\mathbf{x} \;\; \dot{\mathbf{x}}]'$, consisting of its position $\mathbf{x}$ and velocity $\dot{\mathbf{x}}$, and the shape of the potential function U within which it moves can be inferred from a series of noisy observations $y_{1:n}$ at observation times $t_1, ..., t_n$ by applying a Rao-Blackwellized particle filter to the state-space model described above. This idea is similar to that used in the the FastSLAM algorithm of [245] and [230], where a particle filter is used for tracking, and a map (in this case the potential field and in the case of FastSLAM, a set of landmarks) can be efficiently inferred conditioned on the tracks obtained from the particle filter.

5.3.1 Inferring the Potential

Given a series of sampled state estimates $X_{t_1:t_n}$ integration of the motion model (5.1) via the numerical scheme in equation (D.12) can be used to make a 'pseudo-observation' of the potential function U (at positions $\mathbf{x}_{t_1:t_n}$) at each observation time [3]. If values for $\mathbf{x}_t$, $\dot{\mathbf{x}}_t$ and $\mathbf{x}_{t+h}$ are known for some t (as is the case for a single particle in a particle filter), then from the first component of the numerical integration in equation (D.12) it is known that

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\dot{\mathbf{x}}_t + \frac{1}{2}h^2 f(\mathbf{x}_t) + BZ_{2,t} + O(h^3). \tag{5.16}$$

Rearranging this and replacing $f(\mathbf{x}_t)$ with $-\frac{1}{m}\nabla U_{\mathbf{x}_t}$, gives

$$\begin{aligned} -\nabla U_{\mathbf{x}_t} &= \frac{2m}{h^2}\left(\mathbf{x}_{t+h} - \mathbf{x}_t - h\dot{\mathbf{x}}_t\right) - \frac{2mB}{h^2}Z_{2,t} - O(h) && (5.17)\\ &\approx \frac{2m}{h^2}\left(\mathbf{x}_{t+h} - \mathbf{x}_t - h\dot{\mathbf{x}}_t\right) + \epsilon_t && (5.18) \end{aligned}$$

where $\epsilon_t \sim \mathcal{N}\left(0, \frac{4m^2}{3h}\Sigma\right)$. The $O(h)$ term from the integration error has been ignored, since for small h this will be small relative to the $O(h^{-\frac{1}{2}})$ ϵ_t term. This means $(\mathbf{x}_{t+h} - \mathbf{x}_t - h\dot{\mathbf{x}}_t)$ can be treated as an observation of $-\nabla U$ at $\mathbf{x}_t$, distorted by Gaussian noise, which can be incorporated easily into the Gaussian process as noted in section 5.2.2. The observation noise variance scales with $1/h$, meaning that a higher observation frequency does not increases the information available about U (since the number of observations also scales with $1/h$), except in the region where the $O(h)$ term becomes significant (close to $h = 1$). Of course, this only refers to inference of the potential field; more frequent observations are likely to improve the estimation accuracy of the object state. Intuitively, this can be understood by considering that a short motion corresponding to small h will only be deflected slightly by the underlying field, whereas a long motion with large h will suffer more deflection.

5.3.2 Particle filter

To formulate the particle filter it is assumed that a particle representation of the posterior distribution of preceding object states and the corresponding potential field $p(X_{t_1:t_{n-1}}, U \mid y_{1:n-1})$ at t_{n-1} is available, so that

$$\begin{aligned} p(X_{t_1:t_{n-1}}, U \mid y_{1:t-1}) &= p(U \mid X_{t_1:t_{n-1}}, y_{1:n-1}) p(X_{t_1:t_{n-1}} \mid y_{1:n-1}) \\ &\approx \sum_i w_{t-1}^i p(U \mid X_{t_1:t_{n-1}}^i) \delta_{\{X_{t_1:t_{n-1}}^i\}}, \end{aligned} \tag{5.19}$$

where $\delta_{\{X_{t_1:t_{n-1}}^i\}}$ is a delta function at the trajectory of the i^{th} particle $X_{t_1:t_{n-1}}^i$ and where the weights w^i must sum to 1. The distribution $p(U \mid X_{t_1:t_{n-1}}^i)$ is the distribution of the potential field corresponding to the trajectory $X_{t_1:t_{n-1}}^i$ and is given by the Gaussian process approximation, along with the pseudo-observations of the gradient of U derived from the object trajectory $X_{t_1:t_{n-1}}^i$ using equation (5.18). Using these, the value of U can be calculated at any point of interest, and so a sample of U need not be stored in each particle.

When a new observation y_n becomes available at time t_n new samples of the joint state X_{t_n} are drawn from an easy-to-sample proposal density $q(X_{t_n}^i \mid X_{t_1:t_{n-1}}^i, y_{t_1:t_n})$. Given these new samples, the particle approximation of the posterior filtering density can be updated using the standard weight update

$$w_t^{i*} = w_{t-1}^i \times \frac{p(X_{t_n}^i \mid X_{t_1:t_{n-1}}^i)}{q(X_{t_n}^i \mid X_{t_1:t_{n-1}}^i, y_{t_1:t_n})} p(y_n \mid X_{t_n}^i), \tag{5.20}$$

$$w_t^i = \frac{w_t^{i*}}{\sum_i w_t^{i*}}, \tag{5.21}$$

where $p(X_{t_n}^i \mid X_{t_1:t_{n-1}}^i)$ is the state transition density for the model, given by the object dynamics, in this case approximated by the transition density from a numerical integration scheme such as that in equation (D.15). This transition density (and quite possibly also the proposal function) requires the value of the potential field U or its derivatives at a number of points. These can be derived for each particle from the object trajectory for that

particle $X^i_{t_1:t_{n-1}}$, allowing the distribution of the potential field and its derivatives to be evaluated at any points in its domain using the method in section 5.3.1. This allows the transition density to be calculated as

$$p(X^i_{t_n} \mid X^i_{t_1:t_{n-1}}) = \int p(X^i_{t_n} \mid U^i, X^i_{t_1:t_{n-1}}) p(U^i \mid X^i_{t_1:t_{n-1}}) dU^i, \quad (5.22)$$

and similarly for the proposal density q if U is required in forming that. This amounts to Rao-Blackwellization of the Gaussian process approximation of the potential field, using the particle filter mechanism to infer the nonlinear portion of the state consisting of the position and velocity of the object.

In the calculation of $p(X^i_{t_n} \mid X^i_{t_1:t_{n-1}})$ in the Rao-Blackwellized scheme proposed here, the integration over U^i in equation (5.22) requires, in effect, that the necessary values of U^i be treated as a random variables when evaluating $p(X^i_{t_n} \mid U^i, X^i_{t_1:t_{n-1}})$. As these have Gaussian distributions, this is tractable in many cases. For example, section D.3 in appendix D gives transition densities for the numerical scheme in appendix D when the U function is random, as required. This relies on the fact that U is independent of the process noise in the current period, but as noted in section D.4, U depends only on earlier realizations of the process noise and observation noise (via equation (5.18)) and so U and its derivatives will be independent of random variables occurring in the numerical integration scheme (e.g. Z_1, Z_2 and Z_3 in equation (D.12)), which derive from the process noise in the *current* period.

When using a bootstrap filter (in which the state transition density is used as the proposal density), both the process noise variables and the value of ∇U must be sampled to obtain a proposals for the new state sample $X^i_{t_n}$ for particle i at time t_n using the numerical scheme in appendix D. Since, in this case, the derivatives of the potential function are being directly sampled, these samples should be used directly in the inference of the potential field in place of the estimates derived from the subsequent state given in equation (5.18). For multi-step integration schemes this will res-

ult in multiple samples of these quantities along the sample path, which could lead to large numbers of pseudo-observations of the potential field. A practical caveat of using sampled gradients directly is that, because the samples are point samples without any noise, they can lead to numerical instability in the Gaussian process estimation due to badly conditioned covariance matrices. In practice, therefore, it is usually necessary to apply a small amount of artificial Gaussian 'observation' noise to the samples in order to overcome this.

For non-bootstrap filters, proposals can be made from any proposal distribution $q(X_{t_n} \mid U, X_{t_1:t_{n-1}}, y_{t_1:t_n})$, which may be (approximately) adapted to the latest observation y_{t_n}. In order to calculate the weight update in equation (5.20) it must be possible to evaluate both the proposal and state transition densities. For this reason, a numerical integration scheme with a tractable density such as that in equation (D.15) must be used. This need to evaluate transition densities for non-bootstrap schemes can lead to a potential trade-off between such schemes and simulation-only bootstrap schemes, because simulation-only schemes can allow the use of more accurate numerical integration with intractable transition densities; see section D.4. Multi-step schemes can also be used with non-bootstrap schemes, although these requires the use of methods such as sequential imputations to sample intermediate state distributions; see section D.5.

Once a proposal is made, the proposed state $X^i_{t_n}$ can be used to obtain a further noisy pseudo-observation of the force exerted on the target by the potential field corresponding to the proposal using equation (5.18).

5.3.3 Fast covariance updates

The most computationally expensive part of Gaussian process calculations is the inversion of the the $(K + \Sigma_{\text{noise}})$ matrix in equation (5.6) when calculating the process distribution at test points. If K is an $n \times n$ matrix (corresponding to n pseudo-observations of the state), K_{**} is an $m \times m$ matrix (corresponding to m test points) and K_* is a $n \times m$ matrix, then inverting

K is an $O(n^3)$ operation using a naive implementation and $O(n^{2.373})$ in the best known implementation [256]. In general $n \gg m$, since there are likely to be many more observations than test points.

Upon adding p more observations this calculation has to be repeated, so it is desirable to sequentially update this inverse as new observations arrive (in a way that is cheaper than recalculating the inverse itself). This is easier to do if the Cholesky decomposition of $(K + \Sigma_{\text{noise}})$ is stored instead of its inverse. This has the additional advantage that use of the Cholesky factor in place of the matrix inverse is more numerically stable [229]. Here the lower Cholesky factor L is used, which is a lower triangular matrix such that $(K + \Sigma_{\text{noise}}) = LL'$. It is always possible to find this since covariance matrices are positive definite. Given L, the term $(K + \Sigma_{\text{noise}})^{-1}F$ in equation (5.6) (where F is either Z or K_*) can be easily calculated by letting

$$x = (K + \Sigma_{\text{noise}})^{-1}F$$

where x is the required result, so that $F = LL'x$. Writing $y = L'x$ gives $F = Ly$. Since L is lower-triangular these equations can be straightforwardly solved first for y, then for x by forward and back substitution, respectively (repeating this over each column of F when it is a matrix). This operation takes $O(n^2)$ operations when $F = Z$ and $O(n^2 m)$ when $F = K_*$.

Updating L upon arrival of a new observation, amounting to adding a number of rows and columns to the $(K + \Sigma_{\text{noise}})$ matrix, is straightforward. Let $A = (K + \Sigma_{\text{noise}})$ be the covariance matrix prior to the arrival of the new observations. The required matrix is the lower Cholesky factor of

$$A_+ = \begin{bmatrix} A & v \\ v^T & c \end{bmatrix},$$

where v is a $n \times p$ matrix and c is a $p \times p$ matrix, corresponding to adding p new observations (where it would be expected that $p \ll n$). Writing

$\begin{bmatrix} L & 0 \\ b & d \end{bmatrix}$ for the lower Cholesky factor of A_+, the Cholesky decomposition can be written as

$$A_+ = \begin{bmatrix} L & 0 \\ b & d \end{bmatrix} \begin{bmatrix} L' & b' \\ 0 & d' \end{bmatrix} = \begin{bmatrix} LL' & Lb' \\ bL' & bb' + dd' \end{bmatrix},$$

which requires that

$$\begin{aligned} v &= Lb', \\ c &= bb' + dd'. \end{aligned}$$

The first of these can be solved for b by forward substitution in an $O(n^2 p)$ operation and, once b is found, d can be found by taking the lower Cholesky factor of $c - bb^T$ in an $O(p^3)$ operation. This allows the lower Cholesky factor of $(K + \Sigma_{noise})$ to be sequentially updated as more observations become available, with the update having lower computational complexity than inverting the matrix as long as $n^2 p < (n+p)^{2.373}$, which is guaranteed if $p \leq 0.373n$

5.4 Results

In order to test the tracking and map inference in a real-world setting, a single-target tracking test was set up in which the target, a car, performed several laps of the same circular route. This aimed to illustrate the fact that environment structure (in this case road structure) can be learnt by the algorithm, and that this can be used to improve tracking performance. This application is intended to demonstrate the technique and illustrate the way that it could assist with multi-target tracking or path planning applications in situations when some target motion in a structured environment has already been observed.

GPS tracking data was collected for several car journeys around Cam-

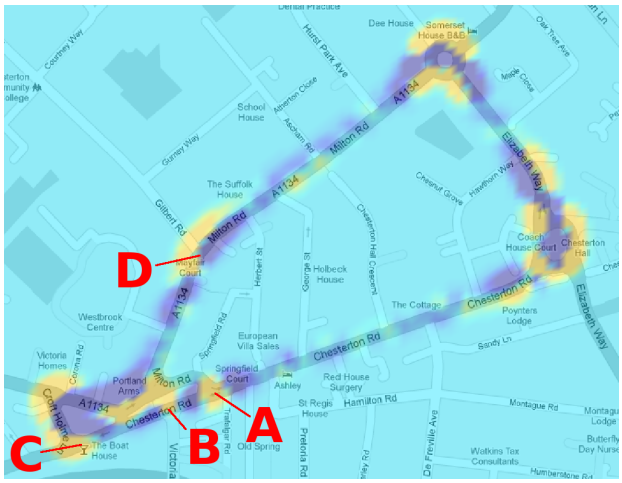


Figure 5.1: Potential field map generated for Mitcham's Corner data using undistorted GPS observations. Dark blue areas are areas of low potential and yellow areas are areas of high potential. The letters A-D show the locations of traffic lights. Maps from Google Maps

bridge, UK. The data was collected using Google's MyTracks app on an HTC Desire Android smartphone. The GPS output seemed to be of high quality, with little noise. This was converted to a 2d position in metres using the latitude and longitude GPS readings (using the local conversion in Cambridge of 111.271km per degree of latitude and 68.372km per degree of longitude). Curvature of the Earth was ignored on this small scale, but for wider area applications this would become an important consideration. It was found that scaling the input data so that characteristic length scales of $l \approx 1$ could be used for the Gaussian process improved numerical stability (for example distances in tens of metres was used with this data).

The dataset "Mitcham's Corner" consists of four loops of about 2.2km through a set of city streets with a variety of road features (traffic lights, roundabouts, junctions). The "Sleaford Street" dataset consists of five loops of a 0.8km route around a set of narrow residential streets with several junctions.

Using the Mitcham's Corner dataset to infer a potential field model for the car's environment gave the results in Figure 5.1. These results show

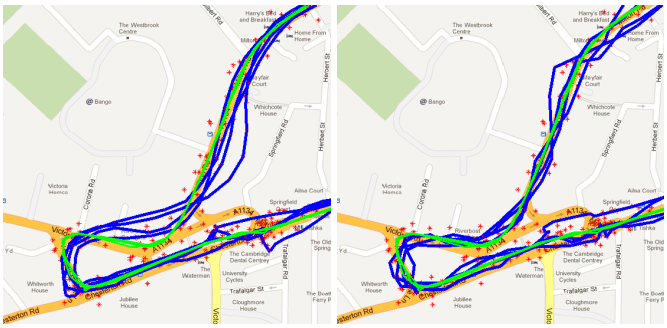


Figure 5.2: Detail of tracking on Mitcham's Corner data (four circuits) without (left) and with (right) map inference. Blue lines show inferred tracks, green lines show ground truth (noise free GPS data) and red stars show generated noisy observations

clear 'channels' (dark blue) in the potential field along roads, with 'hills' in the field where they might intuitively be expected around the edge of corners. The inferred potential map also reveals some hidden features of the road not visible from a standard road map. For example, the traffic lights at A, B, C and D are clearly visible as bumps across the road, due to the speed reduction necessary before arriving at them.

In order to test tracking performance with simultaneous map inference, GPS tracking data was used as ground truth and noisy observations were generated by adding Gaussian noise to the GPS positions according to the model in equation (5.15). The proposed method was then compared to bootstrap particle filtering using the same object dynamics without a potential field (i.e. with $U = 0$ everywhere). The results are shown in Table 5.1. Simultaneous mapping and tracking significantly improves tracking performance. A detail from tracking on the Mitcham's Corner data is shown in Figure 5.2. This shows how, by inferring a potential map, much closer tracking to the detailed road layout was possible.

The system can also be used to establish a map using previous observations (possibly from multiple other object's tracks), which can be used as a prior for subsequent tracking. To test the usefulness of this the last circuit of each of the two routes (subject to Gaussian noise) was used as an

Data	Map & Track	Track only
Mitcham's Corner	12.8 (0.3)	16.7 (1.5)
Sleaford Street	12.6 (0.6)	17.7 (0.6)

Table 5.1: RMS error per observation (in metres) for comparison of simultaneous mapping and tracking and tracking only results. Standard deviations from 10 trials shown in brackets

Data	Perfect	Noisy	None
Mitcham's Corner	13.5 (1.9)	12.4 (0.7)	15.3 (2.2)
Sleaford Street	17.4 (3.4)	14.0 (2.5)	17.8 (4.2)

Table 5.2: RMS error per observation (in metres) for tracking with a range of inferred prior maps. Standard deviations from 10 trials shown in brackets

out-of-sample test path to be tracked, while the previous circuits were used as training data to infer the map. Two versions of this training were tried: one training with the undistorted GPS data ('perfect') and the other with data distorted with the same level of Gaussian noise as the out-of-sample track ('noisy'). The results are shown in Table 5.2. These show that using a map inferred from previous observations improves subsequent tracking. Perhaps surprisingly, training using noisy observations was more effective than with the noise-free GPS observations. This could be because the noise free observations define the roadways on the map too narrowly compared to the noisy data, introducing errors when used with noisy data.

For these tests a bootstrap particle filter with 500 particles was used. The single-step numerical integration scheme in equation (D.12) was used for forward simulation. The standard deviation of the observation noise σ_{obs} was set to between 10m and 30m. The process noise was modelled as independent in each direction (making the $\mathtt{B}$ matrix diagonal) with standard deviation of between 1ms^{-2} and 3ms^{-2}. The characteristic length scale $\mathtt{l}$ for the Gaussian process was set at 15m.

The bootstrap particle filter used here is unlikely to be the best tracking model for this data. However, it does offer a good way to assess the effectiveness of the potential field model. Better tracking algorithms could be developed by using more sophisticated particle filters (for example an aux-

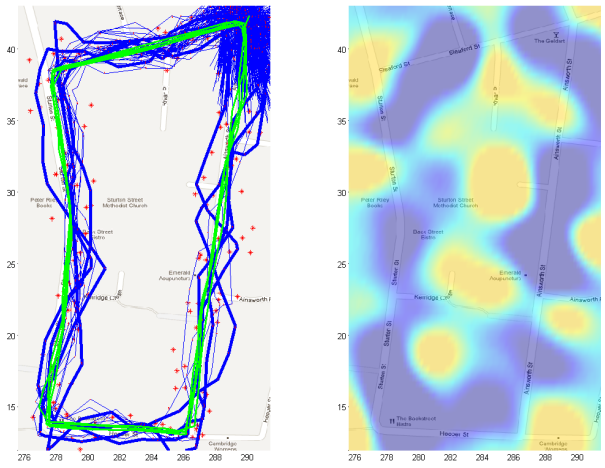


Figure 5.3: Example of simultaneous tracking and potential map inference for Sleaford Street data (five circuits). Lines, points and colours have the same interpretation as in Figures 1 and 2. The scale shows distance in tens of metres. Note the potential channels along roads, with hills at corners. Some road obstacles (parked cars on narrow streets) are visible, for example in the lower right

iliary particle filter with UKF proposals), or by modifying the dynamical model of the tracked object in equation (5.1), to better reflect its true dynamics. For example, for car tracking an intrinsic model that incorporated some knowledge of car dynamics such as their acceleration and braking force ranges could be used.

5.5 Conclusion

This chapter presents a new method of sequentially learning about the structure of an unknown environment while tracking targets moving within it. Unlike existing constrained tracking techniques, no knowledge of the environment structure is necessary in advance, although it is possible to incorporate such knowledge if available. For example, road maps could be used to set the mean of the Gaussian process prior in a similar way to the construction of a likelihood function based on road maps used in [235]. Learnt structural information can be incorporated directly into the tracking

process in order to enhance its performance. The environment is modelled as a potential field map, but only weak non-parametric assumptions need be made about the shape of that field and thus the model is applicable to a wide range of settings. An efficient Bayesian solution method based on the Rao-Blackwellized particle filter is demonstrated. The method can improve tracking performance in structured environments as illustrated by its application to urban car tracking. Map inference can reveal hidden environment structure such as the location of traffic lights, junctions and obstructions in the urban car tracking example here. The method also offers a way of incorporating existing prior knowledge (e.g. from maps or from previous tracks) into a tracking model, which can be updated using feedback from the environment.

The method suffers from the fact that calculations involving the Gaussian process are $O(n^3)$ in the number of observations n, which makes it unsuitable for use with long time series. It also suffers from particle path degeneracy in early stages, which reduces the quality of the map produced. Both of these problems can be overcome, however, and ways to do this are are discussed below.

5.5.1 Further work

There are a number of immediate extensions of this work that could improve its performance and applicability.

In order to increase particle diversity in the particle history, periodic backward sampling can be used in the resampling step of the particle filter using the method of [146]; see section 2.3.3. Since the resampling step aims to redraw samples from $p(X_{t_1:t_n} \mid y_{1:n})$, it is legitimate to use backward sampling there, and this would be particularly beneficial in this case (and other SLAM problems) since the map depends on the entire state history for each particle. Using a back-resampling step allows more diverse particle histories to be produced, which should lead to better map estimates corresponding to early observations. Such a back-resampling step would be

expensive, but would only be necessary periodically, when the past particle history has degenerated sufficiently. MCMC move steps could be also be added to the particle filter (see section 2.3.2) in order to improve particle diversity. It might also be possible to apply some of the recent forward smoothing work in e.g. [163] to create map estimates as forward functionals based on smoothing estimates of the state, though this requires that relatively simple, sequentially updateable sufficient statistics of the map can be devised, which is an open question.

A key improvement to the method would be to use a sparse approximation to the Gaussian process potential field map. There are a number of suitable methods (see, for example, chapter 8 in [229] or [257; 258; 259; 260; 261]). In particular the *Projected Process* (PP) approximation appears to offer a good compromise between error and runtime [229]. This method reduces the computational complexity of calculating the mean and variance of the Gaussian process at a test point to $O(m)$ and $O(m^2)$, respectively, where m is much smaller than the number of observations n. Initialization takes $O(m^2 n)$ time and must be repeated each time a new point is added to the *active set*, the set of m points that define the process, but this need not happen after every observation. An alternative simple method is the *Subset of Datapoints* (SD), in which a subset of the datapoints is chosen and used in the standard way in place of the full dataset. This has $O(m^3)$ initialization complexity, and the same complexity as the PP method for mean and variance calculation. Since these methods do not scale with the number of observations n (or scale only weakly), they can make the method plausible for long time series and continuous online estimation use.

Not all new points need to be added to the active set and several heuristic schemes have been proposed to select points to be added. These are generally greedy schemes, with points being assessed, and the active set being built, dependent on the order in which the points are seen, though some are not, e.g. [257; 261]. Greedy schemes are well suited to sequential applications such as that here and are implemented by evaluating a met-

ric for each new point and adding the point to the active set if this metric is above some threshold. Such metrics include differential entropy score [259], novelty (predictive variance at the observation point) [258] and (approximate) information gain [260].

Figure 5.4 shows a comparison of the SD and PP schemes with the standard non-sparse Gaussian process approach, using a greedy algorithm to select points via the novelty criteria. This shows that the PP scheme in particular offers good replication of the full Gaussian process, at least for this 1d example, while having greatly reduced computational complexity. This makes it an good candidate for a sparse approximation for the Gaussian processes used in this chapter, although the time complexity is proportional to n when adding a new point to the active set, which might be problematic for very long time series with many highly novel observations.

Sparse GP implementations based on heuristic criteria for the information content of observations (such as the novelty criteria, which depends solely on observation position) will be most efficient when many observations lie close together and can therefore be approximated by a few representative observations. Long linear tracks, for example, could still cause computational problems for sparse methods, with each new point containing substantial information and thus being added to the active set. This can be overcome by exploiting the fact that observations in Gaussian processes using the squared exponential covariance function, as here, have weak long-range interaction. Because of this, multiple smaller overlapping Gaussian process 'tiles' could be used to cover the space of interest whilst only introducing a small approximation error. Observations would fall onto at most four of these tiles (at overlapping corners) and the tile whose centre lay closest to the test point would be used for estimation at that test point. The amount of overlap and the tile size could be tuned for the application. On its own this strategy is insufficient to reduce the $O(n^3)$ worst case, since all observations could fall within a single tile. However,

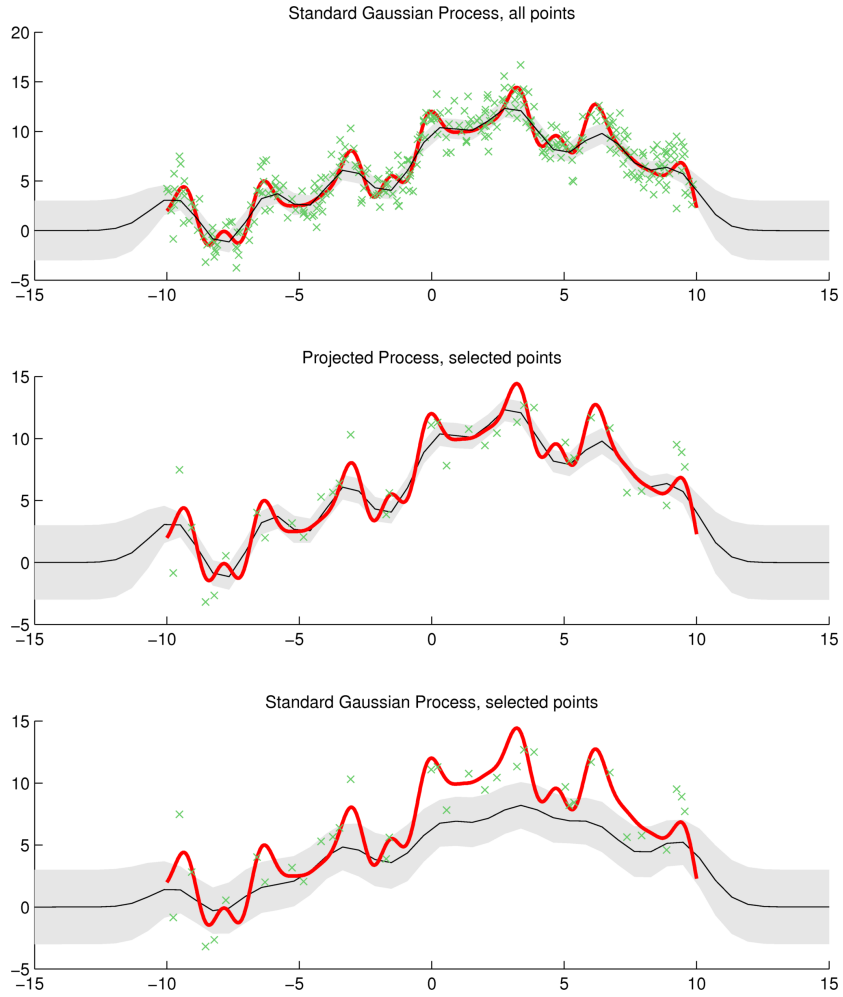


Figure 5.4: Gaussian process approximation showing mean (solid black), and $\pm$ three standard deviations (grey shading) to a function (red) in the interval [-10, 10], based on observations (green crosses) derived from true process with additive Gaussian noise with variance 2. The top panel shows a standard Gaussian process approximation using 300 observations; the middle panel shows the Projected Process approximation using 36 points selected greedily (selected if novelty score was above a threshold of 0.55); the bottom panel shows standard Gaussian process approximation using the same 36 points

combined with sparse GP methods within each tile, the pathological cases of both techniques can be overcome.

Chapter 6

Non-Parametric Group Structure Inference

This chapter examines the problem of Bayesian learning of dynamic network structures for groups of interacting stochastic time series in continuous time, linked by a restricted but useful class of non-linear dynamic Bayesian networks (DBNs). The motivating application for this work is the tracking of multiple interacting physical objects and, in particular, learning about the relationships between them, although the methods could be applied to problems of estimating the interaction of time series, including causal relationships, in other domains. Indeed, related models have been successfully applied to the statistical inference of genetic regulatory networks and financial time series (see section 6.1).

The method presented in this chapter is able to learn the functional form of non-linear relationships between a group of interacting physical objects from a series of noisy observations of their position. This is based on non-parametric assumptions about the form of these interactions, specifically the imposition of a Gaussian process prior on that form. This allows a fully Bayesian solution of the problem, with sampling of the posterior being achieved through an efficient MCMC scheme. This is made possible for long time series by the introduction of a bin-based sparse Gaussian process

approximation, which eliminates the cubic complexity of standard Gaussian process methods with the number of observations and thus allows long series of observations to be used for inference.

Observations can be noisy and can arise from a wide range of observation models, including non-linear and non-Gaussian models. In this chapter, a range of methods are proposed to allow the use of such observations for structure inference, including a simple to implement Gibbs sampler, a sampler based on proposals from bridging distributions to improve mixing, and a Particle Gibbs method able to cope with highly non-linear models. An adaptation of the Particle Gibbs algorithm is presented that allows object trajectories to be sampled individually, helping to overcome curse-of-dimensionality problems that would otherwise hamper inference in systems with many objects.

Interactions between pairs of objects are assumed to occur along vectors derived from their joint state and have strength functionally related to a one dimensional quantity, also derived from that joint state. So, for example, functions of some aspect of object state (e.g. magnitude) or relative state (e.g. distance) can be used; multiple types of relationships can be incorporated within the same framework (e.g. based on magnitude, distance and relative velocity). It is the forms of these potentially non-linear 'linkage' functions that are inferred, with only limited assumptions made about their form. Within this framework many useful systems can be described, including systems of objects connected by springs and dampers, electrostatically or gravitationally connected systems, flocking and coordinated motion models, and systems in which objects avoid collisions.

In this chapter inference is specialized to networks of objects whose relative positions determine their interaction. This describes physical situations in which interaction occurs along the line between two objects, in a distance-dependent way such as those arising from springs, gravity, electrostatics, and collision avoidance.

The method presented here substantially extends that presented in earlier

work in [4]. Some of the key shortcomings of that work have been addressed, in particular the cubic time complexity with respect to the number of observations, and the ability to cope with noisy observations.

The chapter is structured as follows. Section 6.1 surveys related work in network inference and object tracking. Section 6.2 presents the model of object interaction and dynamics, and gives a specific model applicable to distance-based relationships in tracking problems. Section 6.3 shows how a Gibbs sampling scheme can be developed for this model and discusses the algorithmic complexity of this scheme, which corresponds to that presented in [4]. Section 6.4 presents results from this algorithm on synthetic data. Section 6.5 gives details of the sparse bin-based Gaussian process scheme that allows large datasets to be tackled and presents results that suggest this scheme is consistent. Section 6.6 shows how noisy observations can be used for inference and presents a range of schemes for simultaneous state and linkage inference. Section 6.7 presents results of these algorithms, including successful inference of the structure of a group of eight objects linked in a complicated structure. It also gives details of a Particle Gibbs scheme that allows object paths to be sampled individually. Section 6.8 draws conclusions and gives some suggestions for future development of the ideas presented.

6.1 Related Work

Networks in which an object directly influences its neighbours provide a useful description of many situations. They have been used to model observed data in a number of areas including sociology [262] (even before electronic computation), economics and finance [263], computer vision [264], genomics [265; 47; 266; 267] and tracking [142; 268; 269; 270]. In order to gain insight into system structure, inference of the structure of the network has increasingly become a focus of research. In particular, the development of DNA microarrays in the mid 1990s and the resulting

vast increase in the quantity and availability of gene expression data has led to substantial interest in the statistical inference of gene regulatory networks (GRNs). Several approaches have been proposed for modelling such networks, including Boolean networks, e.g. [271], systems of differential equations e.g. [272], and, of particular interest here, Bayesian networks; [273; 266] review these various approaches in the context of GRNs.

Bayesian networks [274] model systems of variables as directed acyclic graphs (DAGs), with nodes corresponding to variables and edges representing causal links between them. In network learning, properties of the objects of interest are represented by the variables (in the case of GRNs, gene expression levels), with the aim being to infer the graph structure that links them. Such network inference can be tackled via a number of approaches, but methods divide into two classes: scoring methods, that aim to find an optimal structure according to some metric of network 'goodness'; and more fully Bayesian methods based on sampling of network structures, that aim to produce a posterior probability distribution over network structures.

Scoring methods such as [275; 265; 276] are usually based on one of a handful of information criteria developed for model selection that aim to balance the model's quality of fit to the data with its parsimony. This is an application of Occam's razor to model selection; see e.g [36] for a discussion of this principle. One common criterion is the Bayesian Information criterion (BIC) [277], equivalent to an approximate penalized likelihood and used, for example, in the structural EM approach of [275]. Another popular criterion is the Bayesian Dirichlet equivalent (BDe) introduced by [276], which is the posterior density of a discrete-valued model using a Dirichlet prior with certain hyperparameters on the transition probabilities between states. Maximizing this metric therefore corresponds to finding the MAP solution under that particular prior.

Sampling methods are generally more demanding [278], but, when they do not become trapped in local posterior minima, have the advantage of

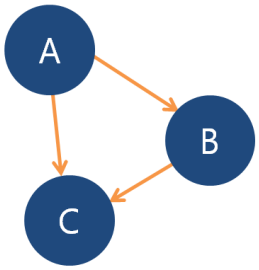


Figure 6.1: Bayesian network representing causal relationships from $A \rightarrow B$, $A \rightarrow C$ and $B \rightarrow C$

giving some idea of the distribution of possible network structures. An early example of such a method is [279], based on defining the proposal step in a Metropolis-Hastings algorithm as a move to a 'neighbouring' network containing one edge more or fewer than the current sample. [267] describes a more recent sampling approach that attempts to address the problem of local minima through a sophisticated MCMC scheme and a parameterization of candidate structures based on the ordering of network graphs by the parent relations between nodes, which is able to reduce the sample space from $O(2^{N^2})$ network structures to $O(N!)$ orderings [280].

A limitation of Bayesian networks is that they are designed to describe static systems and cannot, therefore, account for loops in the graph, such as those arising from feedback mechanisms; see figures 6.1 and 6.2. This has driven the use of dynamic Bayesian network (DBN) models for network inference in GRNs. In these models the system is represented as a set of linked time series, one for each object of interest, generally with a Markovian structure [281]. DBNs have long been used in various branches of signal processing, including tracking (see below), econometrics [282], audio processing [283], vision [264], etc., and correspond to multivariate state space models. State inference, parameter estimation and likelihood evaluation in these models can be tackled if their structure is known using the methods outlined in section 2.1.

Approaches for learning the structure of DBNs are similar to those for Bayesian networks. This was first attempted for GRNs in [284], which

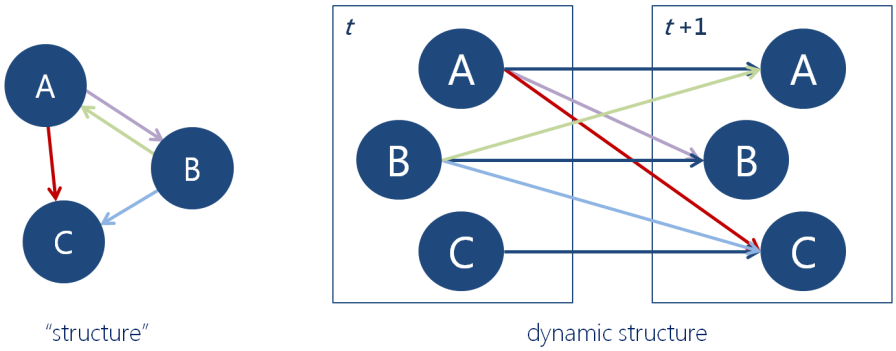


Figure 6.2: Structure (left) with feedback that cannot be represented with a static Bayesian network, and the corresponding dynamic Bayesian network structure (right), shown for two time periods

used a model with discrete-valued variables and applied the BDe scoring criteria in order to find the MAP network structure; [285] and [281] take similar BDe-based approaches. Discrete-valued variables do not always fit real data well [285] and so learning methods for continuous-valued DBN structure have been examined by a number of authors. For linear Gaussian DBNs, several computationally efficient approaches exist. An early example is given in [42], which gives an EM approach for finding the maximum likelihood state transition matrix (and other system parameters) in such a system. Building on this, [47] gives a variational Bayesian approximation to the distribution of the state transition matrix, which is applied to GRN inference. It is also worthy of note that an elegant Gibbs sampler for the elements of the state transition matrix can be derived along the lines of [178].

Learning of non-linear continuous DBNs was first examined in [285; 286], based on earlier similar work on Bayesian networks [287]. The method is based on the assumption that the effect on a gene expression level of all other genes is the sum of non-linear functions of the expression levels of those genes, plus additive Gaussian noise. These non-linear functions are unknown and are estimated using non-parametric spline assumptions about their shape. This approach is similar in outline to that used in this

chapter. Network learning is performed using a scoring criterion, based on a Laplacian approximation of the likelihood function and penalized (via the prior) based on the number of parents of each node. This is similar in spirit to the BIC criterion. [288] extends this to include differential relationships between variables. Recently a more fully Bayesian approach following similar modelling assumptions has been developed by [289], in which an MCMC scheme is used to sample from the network structure and sampled indicator variables are used on links to allow sparsity in network structure to be inferred.

The motivating application of the work in this chapter is multi-object tracking. Since objects in such applications are composed of multiple state variables (e.g. position and velocity in various dimensions) that have known internal dynamics (motion models derived from physical considerations), the dynamic Bayesian networks have a somewhat more complicated structure than those found in GRN problems, as shown in figure 6.3. Methods that attempt to identify the structure between objects have, so far, mainly focussed on group tracking e.g. [290; 291; 269], where groups are densely connected subsets of objects, or have used ad-hoc networks constructed on a frame-by-frame basis using simple measures such as object separation as in [142]. In this latter method, as well as in [136], Markov random fields (MRFs) are used over the current group structure to model assumptions about object behaviour such as collision avoidance. [269] and [292] track groups of targets and make linear assumptions about the relationship between objects within a group using a 'virtual leader' idea, in which the virtual leader is a linear combination of group members. The work of [293; 270] is able to infer true network structures between targets. Interaction between objects is modelled as piecewise linear, with inferred relationships operating at most ranges, but an approximate repulsion term operating at short separations to avoid collisions. The recent and related work of [294] also uses a linear dynamical assumption to model and infer causal relationships between physical objects during tracking. In [293] such tech-

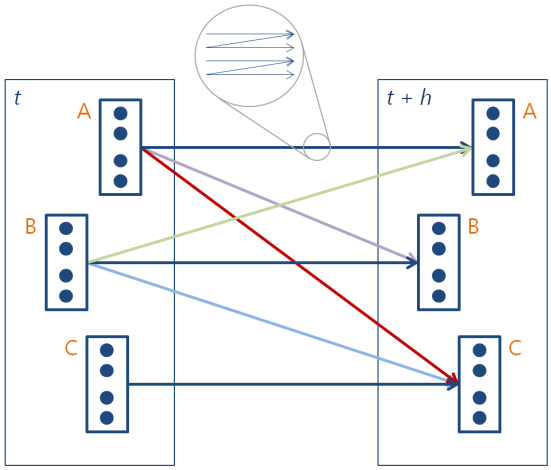


Figure 6.3: Dynamic Bayesian network structure of the type found in physical tracking applications, representing a structure similar to that on the left of figure 6.2

niques are applied to financial time series in order to infer relationships between stock prices and therefore identify suitable candidate stocks for pair trading strategies.

6.2 Model

The model of networked objects employed here is one in which each object in the network has a state consisting of a number of continuous variables, for example position and velocity in Euclidean space. Though it would be possible to extend the framework to discrete or categorical variables this is not considered here. A Markovian time structure is assumed for the system, so that the next state of all objects is determined by their current state (plus noise). The interaction model is based on deriving, for each pair of objects i and j (i.e. from their joint state), some vector Δ_{ij} with dimensionality equal to that of the object state space and some scalar quantity D_{ij}. It must be possible to obtain these deterministically given the joint state of the two objects. The vector Δ_{ij} describes the 'direction' of the interaction between the objects and the scalar D_{ij} is such that a function f_{ij} of this de-

termines the strength of the interaction. The effect on object i from object j is then $f_{ij}(D_{ij})\Delta_{ij}$. The total interaction effect on object i, written F_i, is given by the sum of the interaction effects from all other objects in the system, i.e.

$$F_i(\mathbf{x}_t) = \sum_{j=1}^{N} f_{ij}(D_{ijt})\Delta_{ijt}. \tag{6.1}$$

This aggregate term is the only term for which any sort of observation is possible, and thus the problem is one of determining a large number of individual effects from a series of their aggregate effects observed over time. This is a similar model to that used in [285; 286], although there the per-object effect f_{ij} was simply a univariate function of the univariate state of object j (which can be recreated here by setting $D_{ij} = \mathbf{x}_{jt}$ and $\Delta_{ij} = 1$ for univariate $\mathbf{x}_{jt}$). Since the individual effects at each time are occurring under different conditions, further assumptions about the functional nature of the interactions are necessary in order to relate interactions at one time to those at other times. In this work, an attempt is made to minimize these assumptions through the use of non-parametric Gaussian process (GP) priors.

Object tracking is a motivating example of this work, so this section outlines a concrete object tracking example of this type of problem, in an attempt to motivate and illustrate the approach. The method is applied to interacting systems of physical objects in which inter-object forces depend on the distance between the objects, but this is by no means the only application to which it could be put.

6.2.1 Physical (Langevin) Model

Stochastic systems of multiple objects where inter-object forces depend only on object positions, and in which objects are also subject to random forces of constant variance (for example wind buffeting, thermal noise, etc.), can be written as a particular type of Langevin system, due to Newton's second

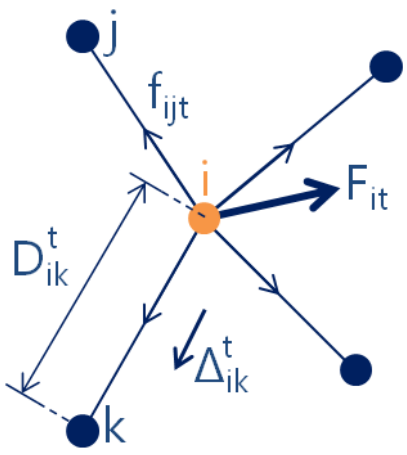


Figure 6.4: The influence of other objects on an object i in a physical model. The arrows show force with f_{ijt} being the force on object i at time t due to object j and F_{it} being the total force from other objects on object i at time t. D^t_{ik} is the distance between objects i and k at time t and Δ^t_{ik} is a unit vector from i to k at time t

law of motion $F = ma$:

$$M d\dot{\mathbf{x}}_t = F(\mathbf{x}_t)dt + B dW_t \tag{6.2}$$

$$d\mathbf{x}_t = \dot{\mathbf{x}}_t dt, \tag{6.3}$$

where $\mathbf{x}_t$ is the collection of the positions of *every* object in the system at time t and $\dot{\mathbf{x}}_t$ is the collection of the velocities of every object at time t. W_t is a collection of the appropriate number of independent Brownian motion noise processes and B is the Cholesky decomposition of the system's noise covariance Σ; it is assumed that this noise covariance does not depend on the system state and is constant over time. In the case of independent noise for each object in each dimension, B will be diagonal, with the diagonal elements corresponding to object i given by s_i, the standard deviation of the state transition density for that object. M is a diagonal matrix of object masses, so that its i^{th} diagonal component is the mass of the object whose state the i^{th} component of $\mathbf{x}_t$ corresponds to. This is similar to the motion model for object in a potential field used in chapter 5.

If the function F is written as a vector of the effects of interaction on each individual object so that

$$F(\mathbf{x}_t) = \begin{bmatrix} F_1(\mathbf{x}_t) \\ F_2(\mathbf{x}_t) \\ \cdots \\ F_N(\mathbf{x}_t) \end{bmatrix},$$

where the F_i are those from equation (6.1), then the motion of an individual object is governed by the second order SDE system

$$\begin{aligned} m_i d\dot{\mathbf{x}}_{it} &= F_i(\mathbf{x}_t)dt + B dW_t \\ d\mathbf{x}_{it} &= \dot{\mathbf{x}}_{it} dt, \end{aligned}$$

where $\mathbf{x}_{it}$ contains the components of $\mathbf{x}_t$ corresponding to the i^{th} object in equations (6.2) and (6.3) and m_i is the mass of the i^{th} object. D_{ijt} is defined as the inter-object distance between objects i and j at time t, and Δ_{ijt} as a unit vector pointing from object i to object j at time t, so that

$$D_{ijt} = \|\mathbf{x}_{jt} - \mathbf{x}_{it}\|_2,$$

and

$$\Delta_{ijt} = \begin{bmatrix} \mathbf{0} \\ \frac{1}{D_{ijt}} (\mathbf{x}_{jt} - \mathbf{x}_{it}) \end{bmatrix}.$$

where $\mathbf{0}$ is a column vector of zeros equal in length to the individual object's position vector $\mathbf{x}_{it}$. This ensures that Δ_{ijt} is the same size as the entire object state and that interaction only affects the velocity components of that state. In this setup $f_{ij}(D_{ijt})\Delta_{ijt}$ represents the force acting on object i due to object j in a direction towards that object (since Δ_{ijt} points from i to j) at time t. The magnitude of the force is given by $f_{ij}(D_{ijt})$, a function of the inter-object distance; it is these f_{ij} functions that are to be inferred.

This setup allows a number of common interactions to be represented in a straightforward way. For example, $f_{ij}(D_{ijt}) = k_{ij}(D_{ijt} - l_{ij})$ gives a linear spring with natural length l_{ij} and spring constant k_{ij}. Gravitational relationships can be represented by $f_{ij}(D_{ijt}) = Gm_i m_j (D_{ijt})^{-2}$, where G is the gravitational constant and m_i and m_j are the masses of objects i and j; electrostatic forces can be represented similarly. Collision avoidance might be encoded as a function which is highly negative when D_{ij} is small and zero elsewhere, creating repulsion at close range. For unconnected pairs of objects $f_{ij}(D_{ij}) = 0$ everywhere.

Given the system state at a discrete set of times T, for example as the output of a tracking algorithm, it is necessary to extract the aggregate effect of object interaction, F_i in equation (6.1) (section 6.6 describes how the algorithm can be used directly with noisy observations). As in chapter 5, it is possible to relate the motion of each object to the force that it experiences due to external factors by inverting the state transition function. As in the previous chapter, integration of the governing equations (6.2) and (6.3) cannot be done analytically for general f_{ij} and so numerical integration is necessary to derive an approximate state transition function. In this chapter, the numerical integration scheme in section D.2.1 of appendix D is used, although other schemes could also easily be considered. This allows F, $\mathbf{x}_t$, $\dot{\mathbf{x}}_t$ and $\mathbf{x}_{t+h}$ to be related as

$$\mathbf{x}_{t+h} = \mathbf{x}_t + h\dot{\mathbf{x}}_t + \frac{1}{2}h^2 F(\mathbf{x}_t) + BZ + O(h^3),$$

where h is the time step size for the numerical integration and Z is a multivariate Gaussian random variable with zero mean and covariance $\frac{1}{3}h^3 I$ (with I being an appropriately sized identity matrix). Rearranging this for F gives

$$F(\mathbf{x}_t) = \frac{2}{h^2}\left(\mathbf{x}_{t+h} - \mathbf{x}_t - h\dot{\mathbf{x}}_t\right) - \frac{2B}{h^2}Z - O(h). \tag{6.4}$$

Writing $\mathbf{y}_t = \frac{2}{h^2}\left(\mathbf{x}_{t+h} - \mathbf{x}_t - h\dot{\mathbf{x}}_t\right)$ as a 'pseudo-observation' of F at time t,

$$\mathbf{y}_t \approx F(\mathbf{x}_t) + \nu_t, \tag{6.5}$$

where $\nu_t \sim \mathcal{N}\left(0, \frac{4}{3h}\Sigma\right)$ is the 'observation' noise. The $O(h)$ term from the integration error has been ignored in this approximation, since if h is small this will make only a small contribution to the observation noise.

6.2.2 Gaussian Process Prior for f_{ij}

If $\mathbf{y}_{it}$ is the part of pseudo-observation $\mathbf{y}_t$ pertaining to the component F_i, then for objects in d-dimensional space, each $\mathbf{y}_{it}$ gives d observations of the sum of N of these f_{ij} functions at a known set of points (the D_{ijt}). In general $d \ll N$, but with sufficient observations in time, information about the f_{ij} can be recovered.

In order to make inference about the f_{ij} possible some assumption about their form is necessary. As in chapter 5, a Gaussian process prior with squared exponential covariance is assumed for each of the f_{ij}, albeit here in one dimension, giving the covariance between the function at two points a and b as

$$\operatorname{cov}(f(a), f(b)) = \exp\left(-\frac{1}{2l^2}\|(a-b)\|_2^2\right), \tag{6.6}$$

where l is a hyper-parameter giving a characteristic length scale for the process. As described in section 5.2.2, this allows the distribution of the function at unknown test points to be evaluated given a series of noisy observations.

6.3 Inference

To reveal system structure, the noisy (pseudo-)observations of the aggregate effect of other objects $\mathbf{y}_t$ must be used to infer the shape of the individual f_{ij} functions. Inference is performed using a Gibbs sampler to infer

the values of f_{ij} each time they occur in an observation of $\mathbf{y}_t$. The value of $f_{ij}(D_{ijt})$, henceforth written as f_{ijt}, must therefore be inferred for all $i, j \in 1...N$ and $t \in T \backslash t_{last}$, where T is the set of observation times of the system (the observation at the final observation time t_{last} is not useable because the system state at t and $t + h$ is required to make a pseudo-observation of $F(\mathbf{x}_t)$, resulting in one fewer observation of F than of the system state).

6.3.1 Overcoming Correlation

If the locations of these pseudo-observations are closely spaced, i.e. the relevant objects are at similar distances for several observations, the correlation between the function values at those points induced by the Gaussian process prior will be very strong, which can lead to slow convergence with Gibbs samplers, as described in section 2.2. In order to make the problem more tractable, new variables f^*_{ijt} can be sampled in place of the f_{ijt}. These are defined as

$$f^*_{ijt} = f_{ij}(D_{ijt}) + \epsilon_{ijt}, \tag{6.7}$$

with $\epsilon_{ijt} \sim \mathcal{N}\left(0, \sigma^2_{ij}\right)$, where σ^2_{ij} is an auxiliary variable that makes the problem easier to solve by introducing artificial noise into the pseudo-observations, thus reducing correlation between these variables and helping the Gibbs sampling scheme converge. If the f_{ij} function is deterministic (as assumed throughout this work), reducing σ^2_{ij} to zero reduced the approximation thus introduced in the posterior to zero, so that the posterior distribution of the f^*_{ijt} variables will be that of the f_{ijt}.

Under the approximation introduced by equation (6.7), equation (6.1) is replaced with

$$F_i(\mathbf{x}_t) = \sum_{j=1}^{N} f^*_{ijt}\Delta_{ijt}. \tag{6.8}$$

Without this approximation and in the presence of closely spaced f_{ijt} vari-

ables the sampler can become trapped in a local posterior maximum where the function is smooth but does not well match the observations. This can be particularly troublesome if no good starting point is known, in which case a starting point of $f_{ij} = 0$ might be chosen, however this initial smooth (but bad) function shape can be difficult to escape.

To get good approximate samples of f_{ijt} requires small or preferably zero values of σ^2_{ij}. In order to achieve this, two schemes can be adopted. One is to treat the σ^2_{ij} as model variables, sampling them as described below. This has the advantage that no particular speed of convergence toward zero must be specified, and the value of σ^2_{ij} can also increase, potentially allowing better mixing. A threshold can be used, so that if σ^2_{ij} ranges widely, only samples for which σ^2_{ij} is below the specified threshold are included in the final set of samples. This bears some similarity to ideas like simulated tempering [97; 98], where a relaxation parameter varies by sample, with final samples only being taken when the least relaxed distribution (the target) is being sampled. On the other hand, under this scheme it might take a long time to generate a sufficient number of samples with low σ^2_{ij}. Alternatively, it might be possible to treat σ^2_{ij} like the temperature in a simulated annealing scheme [37], slowly converging to 0. This would have the advantage that the scheme would reach the required low level of σ^2_{ij}, but requires a "cooling schedule" for reducing σ^2_{ij} to be decided in advance, which might be difficult. There is also a risk that such a scheme would produce biased samples, and this would need to be verified before it could be used. Such fixed-cooling schemes have not been investigated further here.

Sampling σ^2_{ij}

The σ^2_{ij} parameters can be sampled using a Metropolis-within-Gibbs scheme in which each σ^2_{ij} is sampled from its conditional $p(\sigma^2_{ij} \mid f^*, X, \sigma^2_{-ij})$ in turn. With respect to σ^2_{ij},

$$p(\sigma^2_{ij} \mid f^*, X, \sigma^2_{-ij}) \quad \propto \quad p(f^*_{ij} \mid X, \sigma^2_{ij}) p(\sigma^2_{ij}), \tag{6.9}$$

where f^*_{ij} is the set of f^*_{ijt} for all $t \in T$. This is because the observations Y ($\mathbf{y}_t$ at all t) are conditionally independent of σ^2_{ij} given the f^*_{ijt} variables. The first term on the right hand side in equation (6.9) is given by the Gaussian process prior on f_{ij}:

$$p(f^*_{ij} \mid X, \sigma^2_{ij}) = \mathcal{N}\left(0, K + \sigma^2_{ij} I\right)$$

where K is the covariance matrix of the $f_{ij}(D_{ijt})$ terms for all $t \in T$. Unfortunately this expression cannot be easily rearranged to give a distribution in terms of σ^2_{ij} from which to sample and so a Metropolis-Hastings sampler must be used to generate each Gibbs sample from $p(\sigma^2_{ij} \mid f^*, X, \sigma^2_{-ij})$. The choice of prior for σ^2_{ij} in this case is fairly free, though it must ensure that σ^2_{ij} does not go negative and, since small values of σ^2_{ij} are ultimately expected, might reasonably favour these. In this work exponential priors favouring small σ^2_{ij} have been used.

6.3.2 Gibbs sampler for f^*_{ijt}

There are two possible approaches for sampling the f^*_{ijt} variables: site-by-site sampling of each f^*_{ijt} as used in [4], and block sampling of f^*_{ijt} for all values of $t \in T$ together. In what follows, let X represent the entire set of system states ($\mathbf{x}_t$ and $\dot{\mathbf{x}}_t$ at all t), from which all Δ_{ijt}, D_{ijt} and $\mathbf{y}_t$ can be derived as shown above, let Y be the set of all observations, and let S be the set of σ^2_{ij} for all (i, j) pairs. In all that follows, sampling with no relaxation term can be achieved by replacing occurrences of f^*_{ijt} directly with f_{ijt}.

Site-by-Site Sampling

The conditional distribution of a single f^*_{ijt} given all other system variables is given by

$$p(f^*_{ijt} \mid f^*_{-ijt}, X, S) \propto p(Y \mid f^*, X) p(f^*_{ijt} \mid f^*_{-ijt}, X, S), \tag{6.10}$$

where f^*_{-ijt} indicates the set of all f^* variables except f^*_{ijt} and S is the set of all σ^2_{ij} parameters. The second term on the right hand side is given by the Gaussian process prior on f_{ij} so that

$$p(f^*_{ij} \mid X, \sigma^2_{ij}) = \mathcal{N}\left(0, K + \sigma^2_{ij} I\right), \tag{6.11}$$

where f^*_{ij} represents the full set of f^*_{ijt} variables for all $t \in T$. With respect to f^*_{ijt},

$$p(f^*_{ijt} \mid f^*_{ij-t}, X, \sigma^2_{ij}) = \mathcal{N}\left(f^*_{ij-t}; \bar{\mu}, \bar{\sigma}^2\right), \tag{6.12}$$

with

$$\begin{aligned} \bar{\mu} &= K'_*(K + \sigma^2_{ij} I)^{-1} f^*_{ij-t}, \\ \bar{\sigma}^2 &= K'_*(K + \sigma^2_{ij} I)^{-1} K_*. \end{aligned}$$

where f^*_{ij-t} is a vector of the f^*_{ijm} values at all times $m \in T\backslash t$. K is a covariance matrix with its $(m,n)^{th}$ element giving the covariance between $f_{ij}(D_{ijm})$ and $f_{ij}(D_{ijn})$ according to equation (6.6), for all observation times $m, n \in T\backslash t$. K_* is a column vector with its m^{th} element giving the covariance between $f_{ij}(D_{ijm})$ and $f_{ij}(D_{ijt})$ for $m \in T\backslash t$.

The first term on the right hand side in equation (6.10) is given by considering how observations Y are generated with respect to the f^*_{ijt} variable. In the non-symmetric case this is given by

$$p(Y \mid f^*, X) \propto p(\mathbf{y}_{it} \mid f^*, X), \tag{6.13}$$

and in the symmetric case where $f_{ij} = f_{ji}$ it is

$$\begin{aligned} p(Y \mid f^*, X) &\propto p(\mathbf{y}_{it}, \mathbf{y}_{jt} \mid f^*, X) && (6.14) \\ &= p(\mathbf{y}_{it} \mid f^*, X) p(\mathbf{y}_{jt} \mid f^*, X). && (6.15) \end{aligned}$$

The distribution of $\mathbf{y}_{it}$ is given by

$$p(\mathbf{y}_{it} \mid f^*, X) = \mathcal{N}\left(\mathbf{y}_{it}; \sum_{k=1}^{N} f^*_{ikt}\Delta_{ikt}, \frac{4}{3h}\Sigma_i\right), \tag{6.16}$$

where Σ_i is the sub-matrix of the noise covariance Σ corresponding to object i. For generality and notational convenience, it is assumed henceforth that the covariance of $\mathbf{y}_{it}$ is Σ_{it}, so that in this case $\Sigma_{it} = \frac{4}{3h}\Sigma_i$. Then, using identity (A.5) from appendix A, $p(\mathbf{y}_{it} \mid f^*, X)$ can be written with respect to f^*_{ijt} as

$$p(\mathbf{y}_{it} \mid f^*, X) \propto \mathcal{N}\left(f^*_{ijt}; \mu_{it*}, \sigma^2_{it*}\right) \tag{6.17}$$

with

$$\begin{aligned} \sigma^2_{it*} &= \left(\Delta'_{ijt}\Sigma^{-1}_{it}\Delta_{ijt}\right)^{-1} \\ \mu_{it*} &= \sigma^2_{it*}\Delta'_{ijt}\Sigma^{-1}_{it}\left(\mathbf{y}_{it} - \sum_{k\neq j} f^*_{ikt}\Delta_{ikt}\right). \end{aligned}$$

Combining this with the expression in equation (6.12) using identity (A.4) an easy-to-sample expression for $p(f^*_{ijt} \mid f^*_{-ijt}, X, \sigma^2)$ is obtained:

$$p(f^*_{ijt} \mid f^*_{-ijt}, X, \sigma^2) = \mathcal{N}\left(\frac{\sigma^2_{it*}\bar{\mu} + \bar{\sigma}^2\mu_{it*}}{\sigma^2_{it*} + \bar{\sigma}^2}, \frac{\sigma^2_{it*}\bar{\sigma}^2}{\sigma^2_{it*} + \bar{\sigma}^2}\right). \tag{6.18}$$

This Gibbs sampling step is repeated for each of the f^*_{ijt} to give (after a burn-in period) a sample-based estimate of the value of each f^*_{ijt}.

Block Sampling

It is also possible to sample the set of variables f^*_{ij} (containing all f^*_{ijt} variables for $t \in T$) in a single sampling step by noting that

$$p(f^*_{ij} \mid f^*_{-ij}, X, S) \propto p(Y \mid f^*, X)p(f^*_{ij} \mid X, S),$$

with $p(f^*_{ij} \mid X, S)$ given by the Gaussian process prior in equation (6.11) and, with respect to f^*_{ij},

$$p(Y \mid f^*, X) \propto \prod_{t \in T} p(\mathbf{y}_{it} \mid f^*, X) \tag{6.19}$$

in the non-symmetric case or

$$p(Y \mid f^*, X) \propto \prod_{t \in T} p(\mathbf{y}_{it} \mid f^*, X) p(\mathbf{y}_{jt} \mid f^*, X) \tag{6.20}$$

in the symmetric case, with $p(\mathbf{y}_{it} \mid f^*, X)$ given in terms of f^*_{ij} by equation (6.17). Since the noise in $\mathbf{y}_{it}$ is independent of that at other times, $p(Y \mid f^*, X)$ can be written in terms of f^*_{ij} as

$$p(Y \mid f^*, X) \propto \mathcal{N}\left(f^*_{ij}; M, Q\right)$$

with $M = [\mu_{i1*}\ \mu_{i2*}\ ...\ \mu_{iT*}]'$ and $Q = \mathrm{diag}([\sigma^2_{i1*}\ \sigma^2_{i2*}\ ...\ \sigma^2_{iT*}])$. Using identity (A.2) from appendix A and the Woodbury matrix identity, the required conditional distribution of f^*_{ij} can be found as

$$p(f^*_{ij} \mid f^*_{-ij}, X, S) = \mathcal{N}\left(f^*_{ij}; \mu, \Sigma\right), \tag{6.21}$$

with

$$\begin{aligned} \mu &= K^*(Q + K^*)^{-1} M' \\ \Sigma &= K^* - K^*(Q + K^*)^{-1} K^*, \end{aligned}$$

where $K^* = K + \sigma^2_{ij} I$. Though perhaps not immediately obvious, this can be recognized as the joint posterior density of a function at the set of observation locations, using a Gaussian process prior (with covariance K^*), given observations at those locations distorted with additive Gaussian noise of covariance structure Q. Thus, this result can be arrived at by treating the μ_{it*} as pseudo-observations of f^*_{ij} at D_{ijt}, distorted by additive Gaussian

noise of variance σ^2_{it*}, with a Gaussian process prior for f^*_{ij} given by equation (6.11).

In fact, the block sampling can be extended further to the simultaneous joint sampling of all f^*_{ijt} in a similar way. Beyond a certain problem size, however, sampling from such large multivariate normal distributions could become computationally costly as it requires the Cholesky decomposition of a $N^2T \times N^2T$ covariance matrix.

6.3.3 Sampling Process Noise

In some cases the process noise will be unknown and must therefore be sampled. For a full, dense covariance structure Σ this can be achieved by introducing a conjugate Wishart prior distribution on the precision matrix (i.e. Σ^{-1}), although this is not tackled here.

Interesting simpler cases include the case when all objects have independent process noise of the same magnitude in each dimension so that $\Sigma = \text{diag}(\sigma^2_{p_1} I_d, ..., \sigma^2_{p_N} I_d)$, and the case when all objects suffer from the same level of process noise, independent in each dimension, so that $\Sigma = \sigma^2_p I$. In these cases, parameter estimation can be tackled by a standard sampling approach. In the latter case of shared noise variance, for example,

$$p(\sigma^2_p \mid X, Y, f^*, \theta) \propto p(X \mid \sigma^2_p, f^*, \theta) p(\sigma^2_p \mid \theta),$$

where θ is a collection of parameters, $p(\sigma^2_p \mid \theta)$ is a prior for σ^2_p and $p(X \mid \sigma^2_p, f^*, \theta)$ is the state transition density for the full state sequence given the value of σ^2_p, given by

$$p(X \mid \sigma^2_p, f^*, \theta) = p(X_1 \mid f^*, \theta) \prod_{t=1}^{T-1} p(X_{t+1} \mid X_t, \sigma^2_p, f^*, \theta).$$

The one-period state transition densities are given by equation (D.14) using the F in equation (6.1) in the numerical integration. These are non-linear, giving a complicated distribution for σ^2_p. In the absence of alternat-

ive options, this must be sampled using a Metropolis-within-Gibbs scheme with, for example, a random walk proposal. The prior must ensure that σ_p^2 does not go negative, but otherwise can be chosen freely. In this work, an exponential prior has been used that (fairly weakly) favours small σ_p^2, as σ_p^2 is generally not expected to be extremely large.

6.3.4 Structural Sparsity

In the method presented thus far, linkage is estimated between all pairs of objects. In reality, it is likely that there will be no linkage between some pairs of objects. In order to encapsulate this idea an indicator variable $Z_{ij} \in \{0, 1\}$ can be introduced for each inter-object relationship. These indicators are assumed to switch the linkage between objects on or off, so that the inter-object force model from equation (6.8) becomes

$$F_i(\mathbf{x}_t) \approx \sum_j Z_{ij} f^*_{ijt} \Delta_{ijt}. \tag{6.22}$$

The distribution of the Z_{ij} variables is given by

$$\begin{aligned} p(Z_{ij} \mid Z_{-ij}, Y, f^*, X) &\propto p(Y \mid Z, f^*, X) p(Z_{ij} \mid Z_{-ij}, f^*, X) \\ &\propto \prod_t p(Y_{it} \mid Z, f^*, X) p(Z_{ij} \mid Z_{-ij}, f^*, X). \end{aligned} \tag{6.23}$$

The distributions on the right hand side can be evaluated for both $Z_{ij} = 0$ and $Z_{ij} = 1$, allowing the ratio $r = \frac{p(Z_{ij}=1|Z_{-ij},Y,f^*,X)}{p(Z_{ij}=0|Z_{-ij},Y,f^*,X)}$ to be calculated. In this case, $p(Z_{ij} = 1 \mid Z_{-ij}, Y, f^*, X) = \frac{r}{1+r}$ and so Z_{ij} can be drawn from a Bernoulli distribution with this probability of being 1. In the symmetric case, $p(Y_{jt} \mid Z, f^*, X)$ must also be included in equation (6.23), as before.

The prior for Z_{ij}, $p(Z_{ij} \mid Z_{-ij}, f^*, X)$ can, in the simplest case, simply be a fixed prior probability of a non-zero linkage function; this is the prior that has been used in this work where applicable, but other priors imposing structured sparsity are also possible (see, for example, chapter 7). The

distribution $p(Y_{it} \mid Z, f^*, X)$ is given for each value of Z_{ij} by

$$p(Y_{it} \mid Z, f^*, X) = \mathcal{N}\left(Y_{it}; Z_{ij} f^*_{ijt} \Delta_{ijt} + \sum_{k \neq j} Z_{ik} f^*_{ikt} \Delta_{ikt}, \Sigma_{it}\right),$$

and must be evaluated for each t with both $Z_{ij} = 0$ and $Z_{ij} = 1$ giving sparsity estimation an $O(N^2|T|)$ time complexity.

6.3.5 Algorithmic complexity

The sampling process outlined is computationally expensive. Most of the complexity stems from the use of the Gaussian process prior for the f_{ij}. The covariance matrix for the process has $|T| - 1$ rows and making predictions or evaluating likelihoods requires this matrix to be inverted, an $O(|T|^3)$ operation (technically an $O(|T|^{2.373})$ operation using the most efficient method yet proposed [256], although $O(|T|^3)$ in standard implementations). For N objects there are $O(N^2|T|)$ variables in the set of f^*_{ijt}, so this leads to a naive $O(N^2|T|^4)$ algorithm for site-by-site sampling of f^*_{ijt} and an $O(N^2|T|^3)$ algorithm for block sampling f^*_{ij}.

Pre-calculation of some quantities can make sampling tractable for somewhat larger problems. For block sampling, precalculation of $K^*(Q + K^*)^{-1}$ used in equation (6.21) can speed up the algorithm, but does not reduce the algorithmic complexity of each sampling step because Cholesky decomposition of a $|T| \times |T|$ covariance matrix is still required for sampling.

For site-by-site sampling, precalculation of the $K'_*(K + \sigma^2_{ij} I)^{-1}$ terms in equations (6.26) and (6.26) for each i, j and t (since the K matrix is different for each combination of these) leaves the calculation of $\bar{\mu}$ and $\bar{\sigma}^2$ in equations (6.26) and (6.26) as a vector inner product, taking $O(|T|)$ time. The precalculation itself takes $O(N^2|T|^4)$ time but does not need to be repeated for every sample. Since a vector of length $|T| - 2$ is stored for each i,j and t the storage requirements are $O(N^2|T|^2)$ and the sampling time for the full set of f^*_{ijt} is $O(N^2|T|^2)$ (in order to achieve this the sums used in the evaluation of σ^2_* and μ_* must also be precalculated and stored, but these are only

$O(dN|T|)$ where d is the dimensionality of the space). Though the time and particularly the space requirements here are onerous, on modern machines they might be plausible up to e.g. $N = 50$, $|T| = 150$, which would require in the region of several gigabytes of memory.

When sampling the σ_{ij}^2 a $|T| \times |T|$ covariance matrix must be inverted, taking $O(|T|^3)$ time, leading the full sampling of the σ_{ij}^2 to take $O(N^2|T|^3)$ time. Since changing σ_{ij}^2 changes the $K'_*(K + \sigma_{ij}^2 I)^{-1}$ terms, for site-by-site sampling with pre-calculation, these must be recalculated before again sampling the f_{ijt}^*, increasing the overall sampling complexity to $O(N^2|T|^4)$.

An alternative scheme for site-by-site sampling is possible, in which, instead of pre-calculating $K'_*(K+\sigma_{ij}^2 I)^{-1}$, the QR decomposition of the matrix $(K+\sigma_{ij}^2 I)$ is calculated for each i and j. This requires $O(N^2|T|^2)$ storage. During the f_{ijt}^* sampling the quantities needed for the Gaussian process prior can be calculated by removing the row and column corresponding to the relevant time t from this QR decomposition. This update can be computed in $O(|T|^2)$ time (by calculating a series of Givens rotations [295]) and the calculation of the relevant quantities can therefore be completed in $O(|T|^2)$ time (since to sample each f_{ijt}^* a system of the form $Ry = b$ must be solved for triangular R, and several further matrix-vector multiplications must be completed; these are all $O(|T|^2)$) operations). The overall sampling time for the f_{ijt}^* is therefore $O(N^2|T|^3)$. The σ_{ij}^2 sampling then involves recalculating the QR decomposition for each i and j, which has overall complexity $O(N^2|T|^3)$. Thus using this algorithm site-by-site and block sampling have the same complexity. Though the asymptotic complexity of this algorithm is lower than the alternative, it has been found to be slower in the problems tested in section 6.4; see figure 6.9.

6.4 Results I

The algorithm described was tested on synthetic models, generated by known systems simulated using a fourth-order Runge-Kutta integrator with a 0.1s

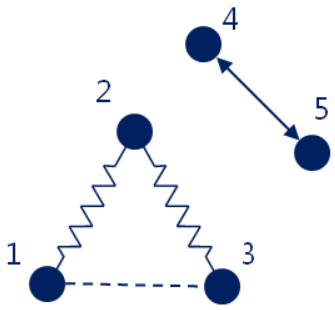


Figure 6.5: System set-up for the results shown in figure 6.6. Zig-zag lines represent linear springs, the dashed line represents a minimum/maximum distance constraint (with force proportional to distance squared beyond these limits) and the double arrow represents an inverse-square attraction between objects

timestep, down-sampled to a 0.2s timestep. Full state information (position and velocity) for all objects was supplied as input from this simulation for 200 samples, corresponding to 40s of data. Length scales for the Gaussian process priors were chosen so that 5 length scales covered the entire distance range for each object pair; this pre-supposes relatively smooth linkage functions in the regions in which they operate, which seems like a sensible prior. Process noise was assumed to be independent for each object in each dimension, corresponding to a diagonal noise matrix Σ, with diagonal elements s_i for object i. The value $s_i^2 = 0.01$ was chosen for all objects as no process noise was introduced in the simulation; 'observation' noise therefore likely derives from the integration error which, being $O(h)$ in equation (6.4), requires s_i^2 to be around $\frac{3}{4}h^3$. For $h = 0.2$ this gives $s_i^2 \approx 0.006$.

Figure 6.6 shows the results for a five object system set up as shown in figure 6.5. These results (and those in figure 6.7) were obtained using site-by-site sampling of f^*, and sampling of σ_{ij}^2; the comparable results for blockwise sampling were indistinguishable. They show four different types of relationship being successfully inferred: linear springs between objects 1 and 2, and 2 and 3; a distance constraint specifying a minimum and maximum separation between objects 1 and 3; an inverse-squared at-

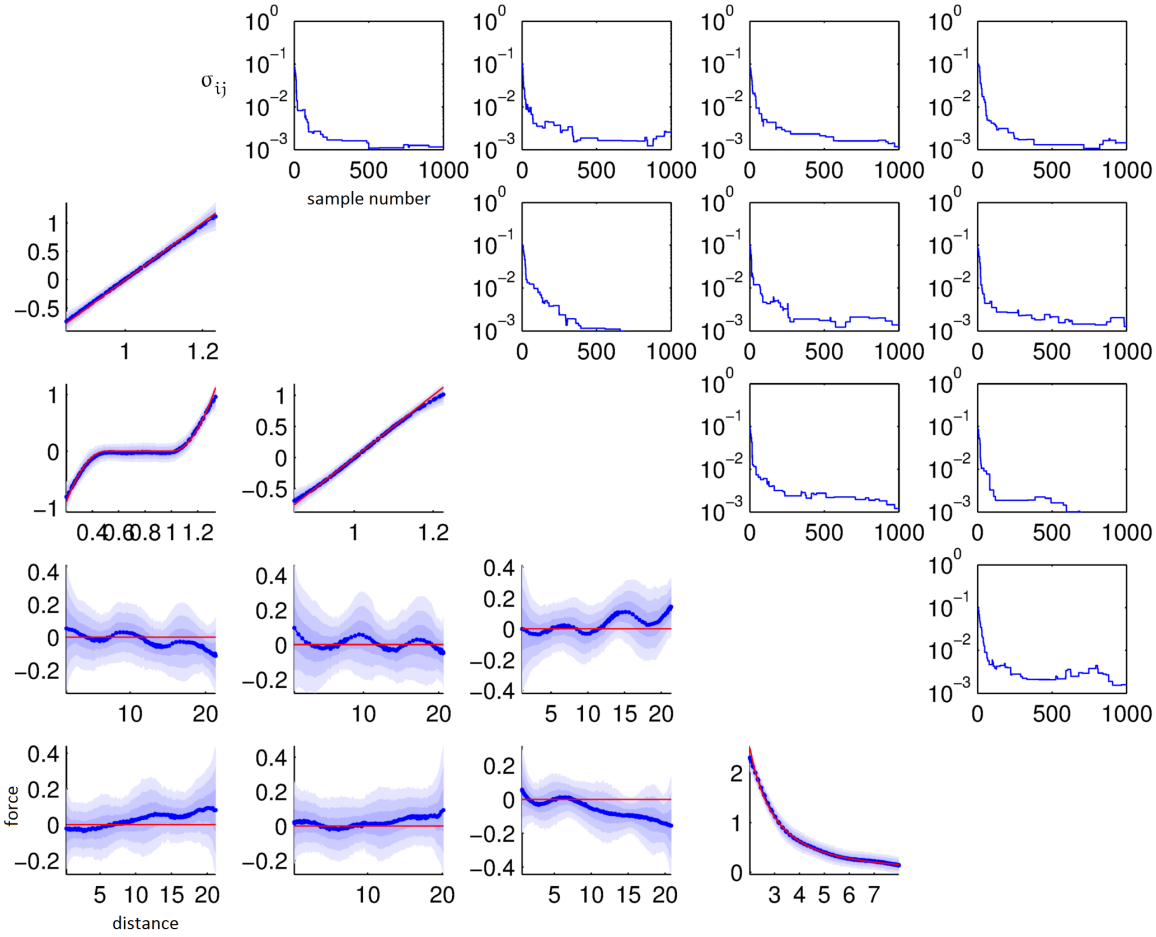


Figure 6.6: Inferred relationships and σ_{ij}'s for a simulation of the symmetric system shown in figure 6.5 with no noise in observations. The graphs in the upper right show the value of σ_{ij} (not σ_{ij}^2) with MCMC sample number (log scale). The lower left graphs show the inferred relationship for each object pair (x-axis is inter-object distance, y-axis is attractive force). In these graphs the red line shows the true relationship, the central blue line shows the mean of the f^*_{ijt} samples for the distance D_{ijt}, with dots showing sample positions (largely indistinguishable here due to the large number of samples); the shading shows the 1, 2 and 3 standard deviation bands. The results are from 700 MCMC samples, after a 300 sample burn-in period. The relationship between objects i and j (for $i < j$) is in row j, column i; the value of σ_{ij} with sample number (for $i < j$) is in row i column j

tractive force between objects 4 and 5; and a zero-force relationship between all other pairs. For the non-zero relationships, the algorithm matches the true relationship very closely. For the zero functions, the algorithm estimates noisy near-zero functions such that zero is within their confidence interval. The bias at extreme points of functions found in [4] was discovered to be due to a coding error, corrected in these results.

The convergence of the MCMC algorithm is indicated by the convergence of the σ_{ij} variables. In figure 6.6 the algorithm converges rapidly (within less than 100 samples). This convergence rate is fairly typical for input with little noise; excessively noisy data can result in very slow convergence, as can misspecification of the noise parameters s_i^2.

Noisy Inputs

Though not specifically designed to cope with noisy inputs, the algorithm can cope with a small amount of noise in its inputs. Figure 6.8 shows the result of applying the algorithm to the noisy position and velocity trajectories shown in figure 6.7. In this test the process noise parameter was increased to $s_i^2 = 0.04$ to attempt to account for the observation noise (this value was chosen by considering the magnitude of the added observation noise and its distortion to the observations given in equation (6.5)). The initial value of the σ_{ij}^2's was also increased to 0.4. The inference of the connecting functions is considerably worse and convergence for the σ_{ij}^2 appears slower, but it is still possible to discern something of the non-zero relationships in the results. As suggested by the choice of noise levels in these examples, the algorithm is much more sensitive to position noise than to velocity noise.

Running Times

Figure 6.9 shows timing results for four versions of the algorithm: site-by-site sampling with both the basic and the QR scheme, blockwise sampling, and a sparse scheme introduced below. All algorithms sampled both f^* and

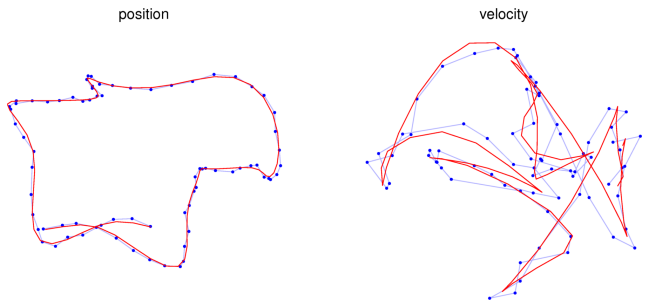


Figure 6.7: Example noisy paths for position (left) and velocity (right) of object 1 in the setup in figure 6.5, shown for 75 timesteps of 0.2s each. The blue dots show observations (linked by pale blue lines) with the red line showing the true trajectory

σ^2_{ij}. They were implemented in a comparable way in Matlab, using pre-calculation to improve speed where possible, and run on a 2009 desktop PC.

The QR version of the algorithm is the slowest to execute but seems to scale with $|T|^2$; scaling with $|T|^3$ is eventually expected, so it is possible that at this scale the algorithm is dominated by an expensive $|T|^2$ process during the update of the QR matrices. The basic site-by-site algorithm works well at medium scales where the $O(|T|^2)$ estimation of f^* took a substantial proportion of time, but $O(|T|^4)$ scaling starts to dominate at longer series lengths as the estimation of σ^2_{ij} and the attendant precalculation begins to take almost all the running time (94% for $|T| = 300$). The blockwise algorithm runs the quickest of the dense algorithms and, at these scales, shows $O(|T|^2)$ scaling (though it is expected that $O(|T|^3)$ effects will eventually dominate and this increase can perhaps start to be detected at longer series lengths). Blockwise sampling can therefore be seen as a significant improvement on the basic site-by-site and QR sampling used in [4], making inference plausible for longer time series. The bin-based sparse algorithm (introduced below) showed by far the best scaling, giving linear scaling with $|T|$. All versions of the algorithm scale with the square of the number of objects.

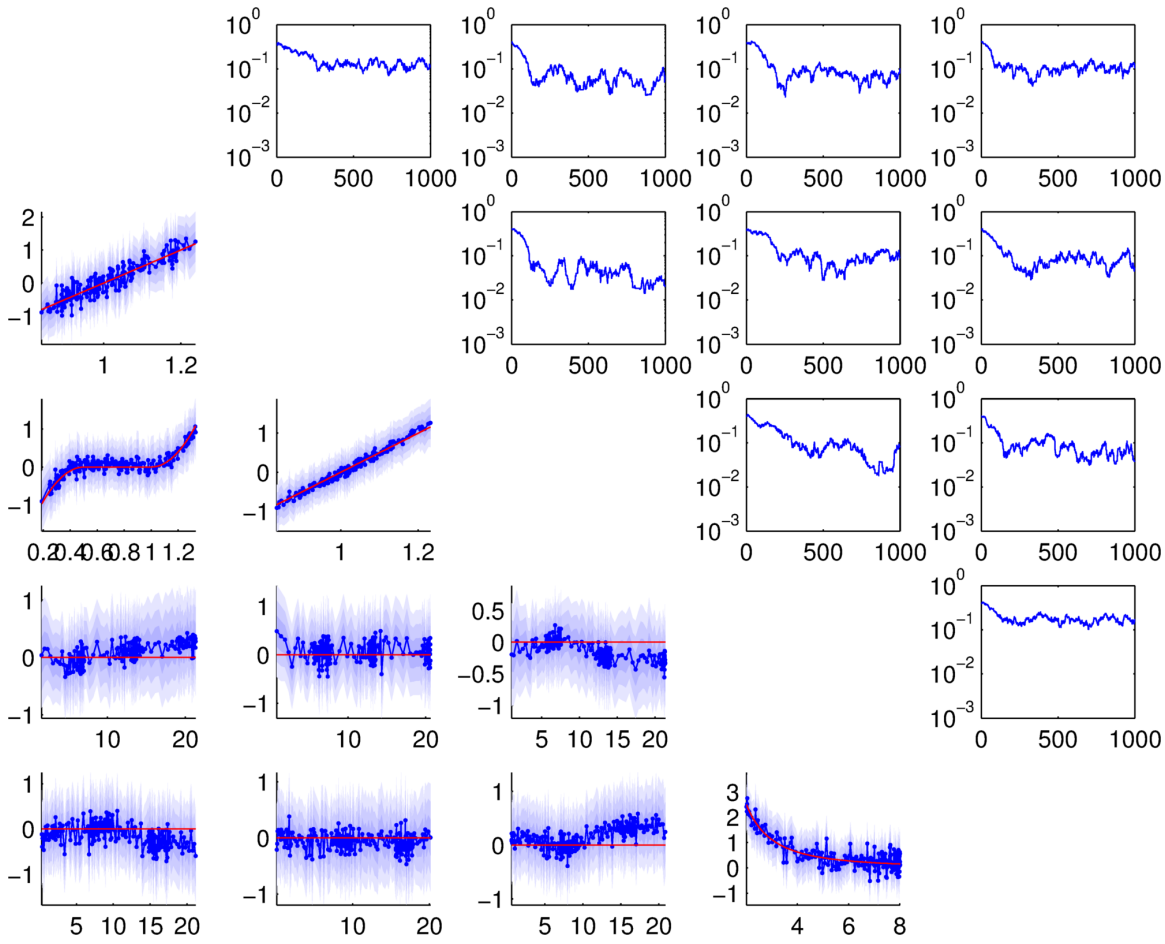


Figure 6.8: Inference and convergence results (similar to those in figure 6.6) for the setup given in figure 6.5 when distorted by additive Gaussian noise such as that shown in figure 6.7. The graphs in the upper right show the value of σ_{ij} (not σ_{ij}^2) with MCMC sample number (log scale). The lower left graphs show the inferred relationship for each object pair (x-axis is inter-object distance, y-axis is attractive force). In these graphs the red line shows the true relationship, the central blue line shows the mean of the f^*_{ijt} samples for the distance D_{ijt}, with dots showing sample positions (largely indistinguishable here due to the large number of samples); the shading shows the 1, 2 and 3 standard deviation bands. The results are from 700 MCMC samples, after a 300 sample burn-in period. The relationship between objects i and j (for $i < j$) is in row j, column i; the value of σ_{ij} with sample number (for $i < j$) is in row i column j

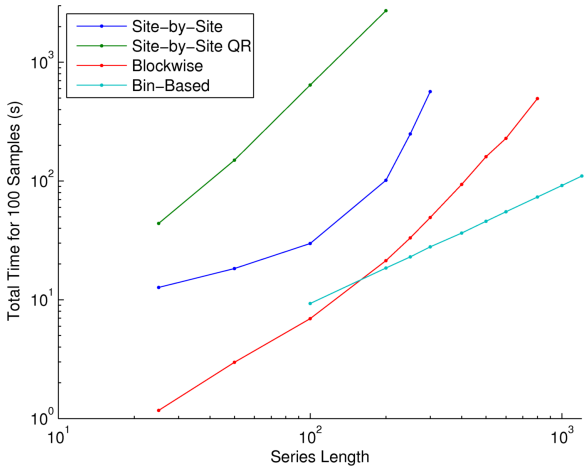


Figure 6.9: Running time for the original and QR variants of the site-by-site sampling algorithm, the blockwise sampling algorithm and the bin-based sparse algorithm (log-log scale) with increasing series length $|T|$

6.5 Sparse Approximation

The method presented so far suffers from high asymptotic complexity; it is simply too slow to be useful for problems of significant size and the fact that it scales with $|T|^3$ means that it is nearly useless for long data series. A major source of this complexity is the Gaussian process prior used for the linkage functions, since with $|T|$ observations the calculation of the prior is $O(|T|^3)$. Conventional sparse Gaussian process approaches such as those discussed in section 5.5.1 are not as helpful in this problem as they are elsewhere. This is because the 'observations' that can be made of the connecting processes f_{ij} are aggregate observations, being the sum over j of all f_{ij} functions at a particular time. This leads to correlation between all observations, resulting in a problem almost as complex as the original. As a computationally tractable alternative, a bin-based approach is introduced in section 6.5.2, where observations are approximated as lying at the centre of one of a finite number of bins. This approach results in a much more efficient algorithm and avoids numerical problems sometimes encountered in the dense version due to the presence of too many observations with little separation, especially with low σ^2_{ij}.

6.5.1 Classical Sparsity

Sparse Gaussian process approximations allow the value of the function in question to be approximated at a given point, often based on a smaller subset of points than the full set of observations. Using these approximations as a replacement for a number of the sampled function values in equation (6.8) leads to a new expression for the total interaction force on object i,

$$F_i(\mathbf{x}_t) = \sum_{j:f^*_{ijt} \in G} f^*_{ijt} \Delta_{ijt} + \sum_{j:f^*_{ijt} \notin G} \tilde{f}_{ij}(D_{ijt}) \Delta_{ijt},$$

where G is the set of M f^*_{ijt} variables that are still to be sampled, with other variables being approximated by the sparse Gaussian process approximation $\tilde{f}_{ij}$. Since the $\tilde{f}_{ij}$ approximations come from different objects j, they can be assumed to be uncorrelated in this sum, so the distribution of the observation $\mathbf{y}_{it} = F_i(\mathbf{x}_t) + \nu_t$ where ν_t is noise as in equation (6.5) can be calculated straightforwardly as a sum of uncorrelated Gaussian random variables. However, with a sparse Gaussian process approximation of f_{ij} the approximation method for $\tilde{f}_{ij}(D_{ijt})$ will depend on the remaining sampled points in G, and, crucially, this will be the case for all t. This means that all observations that depend on the f_{ij} functions (i.e. $\mathbf{y}_{it}$ and, in the symmetric case $\mathbf{y}_{jt}$ for all t) will contain terms that depend on each of the remaining samples f^*_{ijt} in G, and will thus have a dense conditional covariance structure, rather than the block-diagonal structure when all points are sampled. The joint distribution of these $|T| - M$ observations can be calculated and is multivariate Gaussian. Obtaining the pseudo-observation density with respect to each of the remaining sample f^*_{ijt} requires identity (A.5) in appendix A, and this involves calculating the inverse of the corresponding $(|T| - M) \times (|T| - M)$ covariance matrix. Since this is an $O(|T|^3)$ process, there is no overall complexity gain for the blockwise sampler, and only an improvement from $O(|T|^4)$ to $O(M|T|^3)$ for the site-by-site sampler, since there are M rather than T sampled f^*_{ijt} variables. Thus, standard sparse approaches do not provide a useful way of reducing the algorithmic com-

plexity in this case.

6.5.2 Bin-based Sparsity

An alternative approach to reducing the complexity of this system is to approximate each observation of $f_{ij}(D_{ijt})$ as being located at the nearest of M bin centres chosen for the function, i.e. $f_{ij}(D_{ijt}) \approx f_{ij}(\tilde{D}_{ijt})$, where $\tilde{D}_{ijt}$ is the closest bin centre to D_{ijt}. $F_i(\mathbf{x}_t)$ is then approximated as

$$F_i(\mathbf{x}_t) \approx \sum_j f^*_{ijb(t)} \Delta_{ijt}, \tag{6.24}$$

where $b(t)$ is a function (specific for the (i,j) pair) that maps the observation at time t to the nearest bin, and $f^*_{ijb(t)}$ is the 'relaxed' value of f_{ij} at the bin centre nearest to D_{ijt}, i.e. $f^*_{ijb(t)} = f(\tilde{D}_{ijt}) + \epsilon_{ijb(t)}$, where $\epsilon_{ijb(t)}$ takes the same 'relaxation' role as ϵ_{ijt} in equation (6.7), but for the function value at a bin centre. In this setup there are M f^*_{ijp} variables for each (i,j), one for each bin p. For notational convenience $f^*_{ijb(t)}$ will be written as $\tilde{f}^*_{ijt}$, and in general the tilde ($\tilde{}$) decorator will be used to indicate the corresponding value at the nearest bin centre. This approximation corresponds to using a piecewise constant approximation of the function *for regression*, that is for the approximate estimation of the function value at the bin centres from the pseudo-observations. This is equivalent to moving all noisy observations to the nearest bin centre, as illustrated in figure 6.10 and can, in turn, be represented (exactly, in this case) as a single 'observation' at the bin centre with a lower (or equal) noise level. [296] calls this the *compressed* bin approximation.

The idea of binning data to improve computational efficiency is not new and has been examined in detail in the context of local kernel smoothing methods in e.g. [297]. There, several alternative binning schemes are proposed, including the weighted distribution of observations into the nearest two bins, with the relative weight depending on proximity to each. Here, the simple nearest-bin scheme has been used, but other schemes could eas-

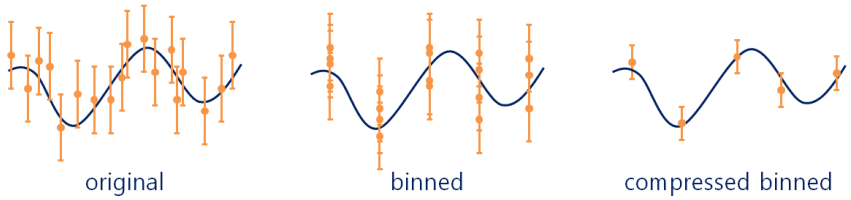


Figure 6.10: Illustration of binning to reduce computation. The original high-noise observations (left) can be approximated by placing them at their nearest bin centre (middle). These collections of observations can be represented as set of lower noise observations, exactly in some cases (including the Gaussian case here)

ily be incorporated, leading to modified calculation of the compressed bin observations.

The number of bins and their location can be varied according to the data. Possible strategies for choosing the bin locations include choosing bin centres so that each bin contains an equal number of observations; choosing centres so that they are uniformly distributed between the limits seen in the data; choosing centres at the location of certain observations based on a greedy novelty criterion; choosing centres randomly, for example through stratified sampling of the interval of observations. Some of these schemes, such as greedy schemes based on observation novelty, could be used in a sequential setting where the observations are not all immediately available. On the other hand, as described in [297], uniformly distributed bins lead to some computational savings because kernel functions (covariance functions in the Gaussian process case) are the same at the same spacing, and there are only a limited number of relative spacings possible on a uniform grid.

In order to calculate distributions for the reduced set of f^*_{ijb} variables (corresponding to noisy function observations at the bin centres), the observation likelihood in equation (6.15) must be recalculated. For the observation $\mathbf{y}_{it}$ where $b_{ij}(t) = b$, equation (6.17) can be replaced with

$$p(\mathbf{y}_{it} \mid f^*, X) \propto \mathcal{N}\left(f^*_{ijb}; \mu_{it*}, \sigma^2_{it*}\right)$$

with

$$\begin{aligned}\sigma^2_{it*} &= \left(\Delta'_{ijt}\Sigma^{-1}_{it}\Delta_{ijt}\right)^{-1} \\ \mu_{it*} &= \sigma^2_{it*}\Delta'_{ijt}\Sigma^{-1}_{it}\sum_{k\neq j}\tilde{f}^*_{ikt}\Delta_{ikt}.\end{aligned}$$

For a particular variable f^*_{ijb}, all the observations in which that variable appears must be considered, i.e. all observations $\mathbf{y}_{is}$ such that $b_{ij}(s) = b$. For symmetrical processes where $f_{ij} = f_{ji}$, the observations $\mathbf{y}_{jr}$ such that $b_{ji}(r) = b_{ij}(r) = b$ must also be included; this is not considered further here, but the modifications to what follows are straightforward. In the non-symmetric case, the overall likelihood expression $p(Y \mid f^*, X)$ is given by the product over all relevant observations so that

$$\begin{aligned}p(Y \mid f^*, X) &\propto \prod_{S_b} p(\mathbf{y}_{is} \mid f^*, X) \\ &\propto \mathcal{N}\left(f^*_{ijb}; \mu_*, \sigma^2_*\right),\end{aligned}$$

where $S_b = \{s \mid b_{ij}(s) = b\}$ is the set of observation times corresponding to bin b. Using identity A.4 from appendix A, the mean and variance are given by

$$\begin{aligned}\sigma^2_* &= \left(\sum_{S_b}\frac{1}{\sigma^2_{is*}}\right)^{-1} \\ \mu_* &= \sigma^2_*\sum_{S_b}\frac{\mu_{is*}}{\sigma^2_{is*}}.\end{aligned}$$

The Gaussain process prior for f^*_{ij} is as in equation (6.11), though in the bin-based method the set of all f^*_{ijx} variables f^*_{ij} is only comprised of f^* values at the bin centres. The covariance matrix K has its $(m,n)^{\text{th}}$ element giving the covariance between $f_{ij}(c_{ijm})$ and $f_{ij}(c_{ijn})$ according to equation (6.6) for all bins $m, n \in B_{ij}\backslash b$, where c_{ijm} is the centre of bin m for function f_{ij}. Similarly, the Gaussian process prior for f^*_{ijb} (for site-by-site sampling) is similar to that that in equation (6.12), with the per-observations quantit-

ies there replaced with per-bin quantities, i.e.

$$p(f^*_{ijb} \mid f^*_{ij-b}, X, \sigma^2_{ij}) = \mathcal{N}\left(f^*_{ij-b}; \bar{\mu}, \bar{\sigma}^2\right), \tag{6.25}$$

with

$$\begin{aligned} \bar{\mu} &= K'_*(K + \sigma^2_{ij} I)^{-1} f^*_{ij-b}, \\ \bar{\sigma}^2 &= K'_*(K + \sigma^2_{ij} I)^{-1} K_*. \end{aligned}$$

Thus f^*_{ij-t} in equation (6.12) is replaced with f^*_{ij-b}, a vector of the f^*_{ijp} values for all bins $p \in B_{ij} \backslash b$, where B_{ij} is the full set of bins used for process f_{ij}. The K_* column vector has its m^{th} element giving the covariance between $f_{ij}(c_{ijm})$ and $f_{ij}(c_{ijb})$ for $m \in B_{ij} \backslash b$. As in section 6.3.2, site-by-site and blockwise Gibbs samplers can be found for the posterior of the function value at the bin centres. The noise level of the f^*_{ij} process given by σ^2_{ij} can be sampled exactly as in section 6.3.1, using the covariance matrix K described.

6.5.3 Consistency and Errors

The bin-based approach generally gives a very good approximation of the full Gaussian process regression with even moderately spaced bins. Figure 6.11 (second panel) shows an example of a bin-based approximation using evenly spaced bins of width equal to the process scale. Here there are 500 observations drawn with Gaussian random noise (top panel), approximated by 40 bins, with only a small difference between the mean of the true Gaussian process and that of the approximation.

There is a limited amount of literature about the quality and consistency of bin-based approximations, but this subject has been examined in the context of kernel density estimates and kernel smoothing, particularly local linear regression, in [298], [299], [296] and [300]. These results can be applied to Gaussian process posterior mean estimation, since that can be cast as a kernel smoother using an *equivalent kernel* k centred on the estimation point x_*, with $k_i = k(|x_i - x_*|/h)$ such that the estimate of $\bar{f}(x_*)$ is given

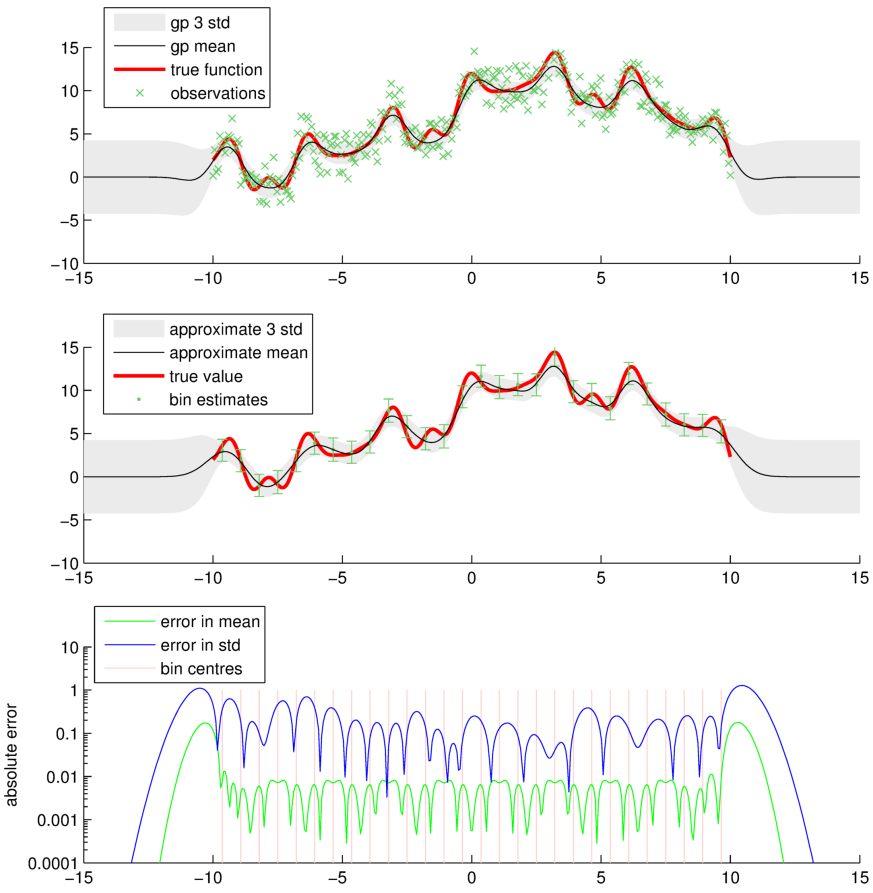


Figure 6.11: Comparison of standard Gaussian process approximation (top) and bin based sparse approximation (middle), with 28 evenly spaced bins of width l (GP length scale). Bottom chart shows absolute error between true GP and bin-based approximate GP, for both mean and standard deviation (log scale)

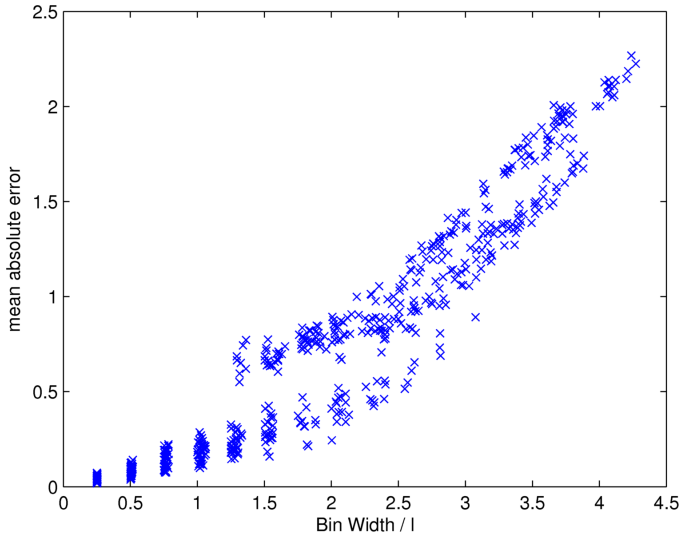


Figure 6.12: Mean absolute error between true Gaussian process and bin-based approximation, against ratio of bin width to l for evenly spaced bins over data range, using function shown in figure 6.11 with various length scales l from 0.5 to 1.5 and between 10 and 500 observations per test

by

$$\hat{f}(x_*) = \sum_i w_i y_i$$

where $w_i = k_i / \sum_i k_i$, $W = [w_1\ w_2\ ...\ w_N] = K(K + \sigma^2_{obs} I)^{-1}$, and h is a smoothness parameter (or bandwidth), which is proportional to l in the Gaussian process case with squared exponential covariance (see [229], section 7.1). For Gaussian processes with the squared exponential covariance structure, the equivalent kernel cannot be given in closed form, but an approximation is given in sections 2.6 and 7.1 of [229]; the specific form of this equivalent kernel is not important in what follows, merely its existence.

The Gaussian process mean estimate is therefore a Nadaraya-Watson estimator (i.e. a kernel estimator approximating the function as a constant at each point) and [299] gives an approximate asymptotic result for binned versions of such estimators uniformly spaced bins. This results shows that the approximate asymptotic mean squared error (AMSE) of the binned es-

timator decreases to zero as $h \to 0$ and is thus (at least approximately) a consistent estimator of the regression model (i.e. of the function f of interest). [300] shows that binned versions of local linear regression estimators with non-negative kernels are exactly consistent (under similar conditions to [299]) as $\Delta \to 0$ and furthermore suggests that the approximate results in [299], "should hold exactly under tighter conditions". It is therefore seems possible that such results could be adapted to show consistency for binned Nadaraya-Watson estimators.

The studies in [300] and guidance in [299] suggest that, in practice, binned estimators are not terribly sensitive to the bin width and that $\Delta/h \approx 0.3 - 0.5$ should work well. This is borne out by the limited study shown in figure 6.12 for the problem in figure 6.11, which shows how the error scales with bin spacing relative to the length scale of the Gaussian process being approximated. In this case, bin widths below l seem to result in low-error approximations.

6.5.4 Prediction

As will be seen below when dealing with noisy observations, the value of inter-object linkage functions (and their gradients) at distances other than those sampled is sometimes necessary, for example when simulating forward or evaluating transition densities. Given the series of samples at the specified distances, several options are available for obtaining the function value at intermediate points.

If the required distances are known when sampling the f^*_{ij}, they can be included in the set of points to be sampled and their value can be found using the Gaussian process approximation. If gradients of f^*_{ij} are required they can also be incorporated using the relations given in section 5.2.2. This method is only appropriate for a small number of additional points; if many points are required the complexity of the Gaussian process approximation will rapidly become excessive due to its cubic scaling.

If the distances are unknown when sampling f^*, or if there are many of

them, interpolation methods are required. The simplest option is piecewise constant interpolation, in which the function value at the nearest bin centre is used. Linear interpolation is perhaps a better alternative, and other interpolation methods such as splines, or further Gaussian processes can also be used. Throughout this work, linear interpolation is used.

If function gradients are required, both the function values and the gradients at the bin centres (i.e. f'^*) should be sampled during the sampling of f^*, using the covariance relationship between a Gaussian process and its gradient given in section 5.2.2 to incorporate them in the set of sampled variables. Interpolation can be used with these to obtain the gradient at the required points. These estimates are much better than those obtained through finite difference approximations derived from the sampled f^* values, especially in the presence of noise. Figure 6.15 shows the results of sampling some function gradients in this way. It can be seen that these estimates have much higher variance than the function estimates, even with noise-free observations. They also suffer more strongly from edge effects, a common problem with many kernel estimators [299]. The quality of the estimates is also more sensitive to noise in the input sequence.

6.5.5 Bin-based Sparsity Results

Figures 6.13 and 6.14 show comparable results to those in figures 6.6 and 6.8 using bin-based sparse estimates in place of dense Gaussian process estimates. The results correspond closely to the dense estimates, suggesting that the bin-based approximation is a good one. The results in figures 6.13 and 6.14 were produced using site-by-site sampling, with ten bins per object pair relationship, since as with the dense process, length scales for the Gaussian process priors were chosen so that 5 length scales covered the entire distance range for each object pair, giving bin spacing of $0.5l$.

Combining the bin-based sampler with blockwise sampling has a major advantage over the dense sampler, however, because the greater spacing of the sample points reduces their covariance, allowing bigger sampling

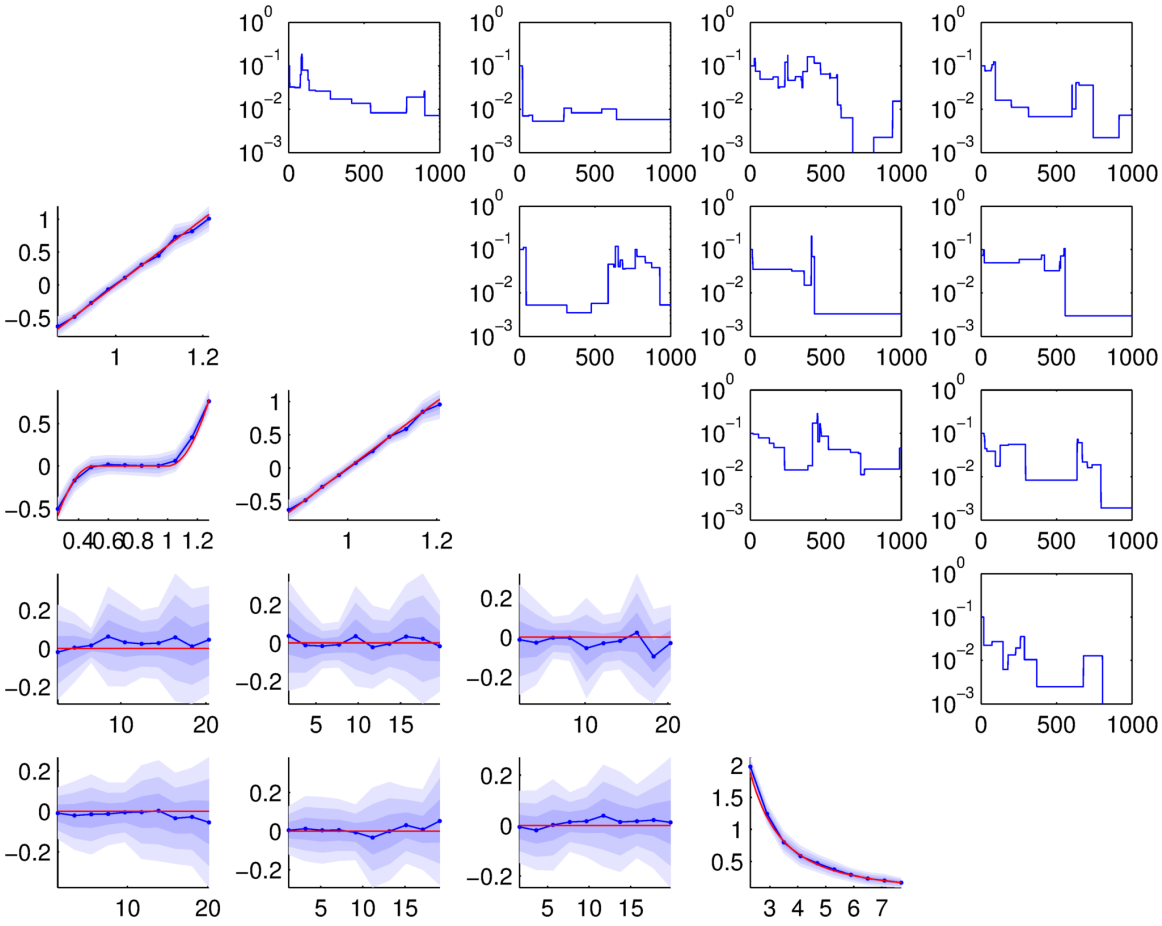


Figure 6.13: Inference and convergence results (comparable to those in figure 6.6) for the setup given in figure 6.5, using the bin-based sparse Gaussian process approximation with 10 bins per object pair. The graphs in the upper right show the value of σ_{ij} (not σ_{ij}^2) with MCMC sample number (log scale). The lower left graphs show the inferred relationship for each object pair (x-axis is inter-object distance, y-axis is attractive force). In these graphs the red line shows the true relationship, the central blue line shows the mean of the f^*_{ijb} samples at the centre of each bin b, with dots showing bin centres; the shading shows the 1, 2 and 3 standard deviation bands. The results are from 700 MCMC samples, after a 300 sample burn-in period. The relationship between objects i and j (for $i < j$) is in row j, column i; the value of σ_{ij} with sample number (for $i < j$) is in row i column j

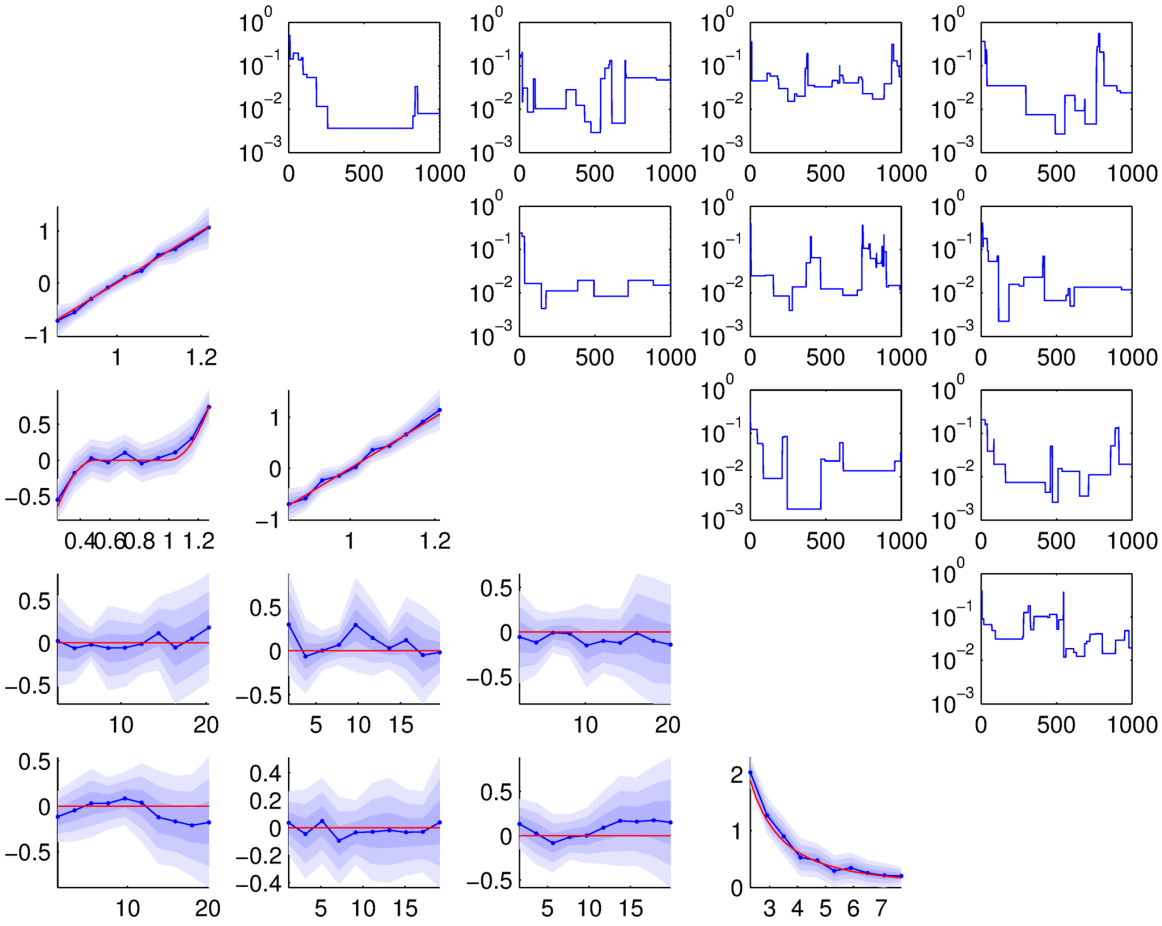


Figure 6.14: Inference and convergence results (comparable to those in figure 6.8) for the setup given in figure 6.5 distorted by additive Gaussian noise (with trajectories as in figure 6.7), using the bin-based sparse Gaussian process approximation with 10 bins per object pair. The graphs in the upper right show the value of σ_{ij} (not σ_{ij}^2) with MCMC sample number (log scale). The lower left graphs show the inferred relationship for each object pair (x-axis is inter-object distance, y-axis is attractive force). In these graphs the red line shows the true relationship, the central blue line shows the mean of the f^*_{ijb} samples at the centre of each bin b, with dots showing bin centres; the shading shows the 1, 2 and 3 standard deviation bands. The results are from 700 MCMC samples, after a 300 sample burn-in period. The relationship between objects i and j (for $i < j$) is in row j, column i; the value of σ_{ij} with sample number (for $i < j$) is in row i column j

moves. Combined with the block sampler which also allows bigger sampling moves, this allows the algorithm to converge to the correct solution *without* the use of the relaxation provided by σ_{ij}^2 in many cases, even from the starting position $f = 0$. Since sampling σ_{ij}^2 is a time consuming component of the previous algorithm, this greatly speeds up computation and removes the need to consider either f^* samples as approximations of the true linkage functions f, or only consider f_{ij}^* samples corresponding to small values of σ_{ij}^2. Running times for the bin-based sampler are shown in figure 6.9, and show that, as expected, the running time of the algorithm does indeed scale with $|T|$, making it plausible for very long data series (for example, it takes about 1s per sample for a series of 1200 observations with five objects). Results of such sampling are shown in figure 6.15 and show good convergence with results comparable to those in figures 6.13 and 6.14, without the need for sampling of σ_{ij}^2.

6.6 Noisy Observations

A weakness of the algorithm thus far is its limited ability to cope with noisy observations of the object positions and velocities. In most scenarios, only noisy observations will initially be available; with the algorithm presented thus far these require pre-processing via another state inference algorithm, which would be unable to take account of any inferred structure. It is therefore desirable to simultaneously infer object states and inter-object structure. This may lead to improved state estimates if interaction is a significant factor in object motion. This section shows how the algorithms described so far can be extended to require only noisy observations as their inputs. There are several possible approaches for this, including standard Metropolis-within-Gibbs for sampling individual states (section 6.6.1), blockwise sampling using bridging distributions (section 6.6.2) and the Particle Gibbs algorithm of [33] (described in section 4.3 and covered for this application in section 6.6.3).

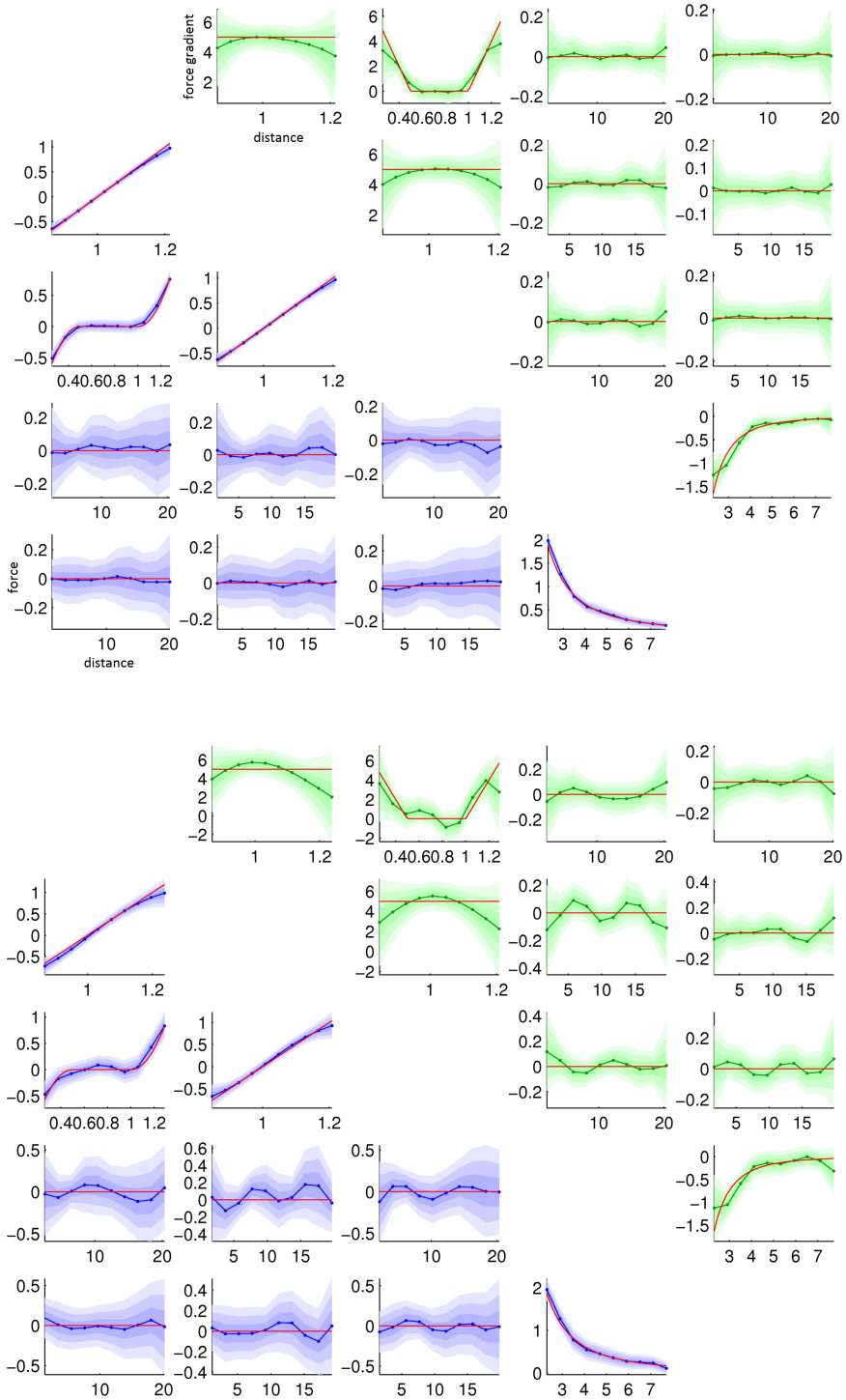


Figure 6.15: Inference results using block sampling of f_{ij} (blue) and f'_{ij} (green, transposed positions) with $\sigma^2_{ij} = 0$ for all (i, j) with clean signal (upper panel) and noisy signal as in figure 6.8 (lower panel). In all graphs x-axis is inter-object distance and y-axis is function value (f_{ij} or f'_{ij}); red lines indicate true values. Results based on 700 samples, after a 300 sample burn-in

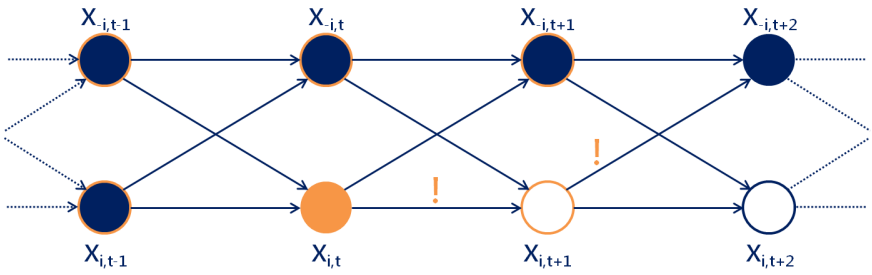


Figure 6.16: State transition structure (discrete or single step integrated), showing Markov blanket for $X_{i,t}$ (orange outline) and conditioning information (solid blue circles) when calculating forward transition density $p(X_{i,t} \mid X_{-i,1:T}, X_{i,t-1})$. Exclamation marks show some links making this conditional density intractable

6.6.1 State Inference via Gibbs Sampling

The simplest approach to sampling the joint object state is with a Metropolis-within-Gibbs algorithm in which the state for each object at each time $X_{i,t} = (\mathbf{x}_{i,t}, \dot{\mathbf{x}}_{i,t})$ is sampled in turn from the conditional distribution

$$\begin{aligned} p(X_{i,t} \mid X_{i,-t}, X_{-i,1:T}, y_{1:T}) &= p(X_{i,t} \mid X_{i,t-1}, X_{i,t+1}, X_{-i,t-1:t+1}, y_t) \\ &\propto p(y_t \mid X_t) p(X_{t+1} \mid X_{i,t}, X_{-i,t}) p(X_{i,t} \mid X_{i,t-1}, X_{-i,t-1}), \end{aligned} \quad (6.26)$$

where conditioning on the sampled interaction function f^* and other parameters is not shown for notational brevity. This can be seen from the problem structure shown in figure 6.16. If the observations are conditionally independent given the individual object states, i.e. if $p(y_t \mid X_t) = \prod_j p(y_{j,t} \mid X_{j,t})$, as when the objects are independently observed, then the observation density in equation (6.26) is given by $p(y_t \mid X_t) \propto p(y_{i,t} \mid X_{i,t})$.

The state transition densities appearing in equation (6.26) are determined by the system model, for example that given in equations (6.2)-(6.3). In general, this is nonlinear and its transition density over a finite period is intractable, requiring a numerical approximation to evaluate. Some such approximations are discussed in appendix D, but use of any of the available numerical schemes requires evaluation of $F(\mathbf{x}_t)$, the function describing the

interaction force on each object due to all other objects; some, such as the higher-order single step scheme of section D.2.1, also require its Jacobian J. These quantities can be evaluated from the estimated f_{ij} functions; the component of $F(\mathbf{x}_t)$ relating to the i^{th} object is given by equation (6.1). The value of $f_{ij}(D_{ijt})$ can be approximately found from the samples of f^* using one of the interpolation methods in section 6.5.4. The Jacobian of F is given by

$$J_{kj|t} = \begin{cases} \sum_i \frac{1}{D_{kit}} \left(\Delta_{kit}\Delta'_{kit} - I\right) f_{kit} - \Delta_{kit}\Delta'_{kit} f'_{kit} & \text{if } k = j \\ \frac{1}{D_{kjt}} \left(I - \Delta_{kjt}\Delta'_{kjt} - I\right) f_{kjt} - \Delta_{kjt}\Delta'_{kjt} f'_{kjt} & \text{if } k \neq j \end{cases} \quad (6.27)$$

where $J_{kj|t}$ is the $d \times d$ sub-matrix of J_t corresponding to $\left.\frac{\partial F_k}{\partial x_j}\right|_{\mathbf{x}_t}$, where d is the dimensionality of the space in which the objects exist. In this expression, $f_{ijt} = f_{ij}(D_{ijt})$ and f'_{ijt} is the gradient of f_{ijt} at D_{ijt}. This latter can be found using an interpolation method from section 6.5.4 applied to the function gradient samples f'^*, which should be sampled simultaneously with f^*. The calculation of J takes $O(N^2)$ time, but J is straightforward to update in place when the state of any object is altered. Changing the state of object i at time t changes f_{ijt}, f'_{ijt}, D_{ijt} and Δ_{ijt} and their opposites f_{jit} etc., meaning that all terms involving these, i.e. $J_{ik|t}$ (and $J_{ki|t}$ in the symmetric case) for all k, must be recalculated. This can be done in $O(N)$ time, meaning that Jacobian updates corresponding to the updating of all object positions can be completed in $O(N^2)$ time.

To update the state of object i at time t, a proposal is drawn from a proposal distribution $q_{it}(X^*_{i,t} \mid X^{cur}, y)$ and accepted with probability

$$p_{accept} = \min\left(1, \frac{p(X^*_{i,t} \mid X^{cur}_{i,-t}, X^{cur}_{-i,1:T}, y) q_{it}(X^{cur}_{i,t} \mid X^*_{i,-t}, X^{cur}_{-i,1:T}, y)}{p(X^{cur}_{i,t} \mid X^*_{i,-t}, X^{cur}_{-i,1:T}, y) q_{it}(X^*_{i,t} \mid X^{cur}, y)}\right), \quad (6.28)$$

where X^{cur} is the current sample of the state of all objects. Three different proposal mechanisms are examined here: random walk, predictive and

adapted. The random walk proposal is the simplest, with

$$q_{it}^{rw}(X_{i,t}^* \mid X^{cur}, y) = X_{i,t}^{cur} + \eta,$$

and $\eta \sim \mathcal{N}\left(0, \Sigma_{prop}\right)$. The covariance of the proposal can be chosen to match roughly the scale of the expected motion over one time period according to the constant velocity model [204]. So, the variance of η is set to $\frac{1}{3}kh^3$ in components corresponding to object positions and to kh in components corresponding to object velocities, with k a constant chosen to give good results. This proposal is symmetric and so cancels in the ratio in equation (6.28).

If interaction effects are strong, it might be more successful to make proposals that take these into account by proposing from the numerical approximation of the model prediction using

$$q_{it}^{pred}(X_{i,t}^* \mid X^{cur}, y) \sim \mathcal{N}\left(\mu_{pred}, \Sigma_{pred}\right),$$

with μ_{pred} and Σ_{pred} given by the state transition function or an approximation, e.g. those of the numerically approximated state transition density from equation (D.14) for a single object. Since this proposal does not depend on the current sample, it leads to an independence sampler.

If the observations are highly informative and the state transition model is subject to a lot of noise, predictive proposals are likely to perform poorly. In this case, an 'adapted' proposal can be used to take account of the observation. This can be done using the 'correct' step from the Kalman filter update, so that

$$q_{it}^{adapt}(X_{i,t}^* \mid X^{cur}, y) \sim \mathcal{N}\left(\mu_{adapt}, \Sigma_{adapt}\right),$$

with

$$\begin{aligned} \mu_{\text{adapt}} &= \mu_{\text{pred}} + K(y_{i,t+h} - H\mu_{\text{pred}}), \\ \Sigma_{\text{adapt}} &= (I - KH)\Sigma_{\text{pred}}, \\ K &= \Sigma_{\text{pred}} H'(H\Sigma_{\text{pred}} H' + \Sigma_{\text{obs}}), \end{aligned}$$

assuming that the observation function is linear with additive Gaussian noise so that $y_{i,t} = HX_{i,t} + \nu$ with $\nu \sim \mathcal{N}(0, \Sigma_{\text{obs}})$. For non-linear observations functions, the corresponding step from the extended Kalman filter (EKF) could be used.

A mixture of these proposals can be used to attempt to improve mixing and their ratio, along with the variance of the random walk proposal, can be adaptively updated during the (fixed-length) burn-in phase of the chain in order to achieve good acceptance rates; see section 2.2. (Finite adaptation leaves the stationary distribution unchanged).

In cases where the model noise is low, such site-by-site sampling can run into difficulties because object states are strongly correlated with the preceding and subsequent object states. This results in poor mixing because the sampler can only move one site (i.e. the state of one object at one time) at a time and thus only small moves are likely be accepted, leading to slow mixing. One possible solution to this problem is to use more sophisticated sampling methods such as simulated tempering or multiple coupled chain methods (see section 2.2.3) making use of a series of less sharply peaked intermediate distributions, throughout which it is easier to move. Initial experiments with the simulated tempering method in [98] gave poor results, with the method requiring substantial tuning to produce any useful samples. The related equi-energy sampler of [301; 267] might offer an alternative, but has not been not investigated further.

6.6.2 Bridge Proposals

To attempt to overcome the slow mixing that can be encountered with site-by-site sampling, larger blocks of states can be sampled. Tractable blocking structures (to numerical approximation) include those in which the state of all objects between two times t_1 and t_2 form a block, although such blocks can be of high dimensionality at each time, potentially leading to low acceptance probabilities and high rejection rates. The problem structure shown in figure 6.16 means that sampling only the states of a single object i (or subset of objects) is only possible by evaluating the full transition density across all objects, since the conditional $p(X_{i,1:T} \mid X_{-i,1:T})$ is generally intractable when the transition density is non-linear.

Good block proposals can be made from distributions that approximate the conditional target distribution in a block $p(X_{t_1:t_2} \mid X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2})$. Such proposals will naturally form a 'bridge' between the states X_{t_1-1} and X_{t+1}, which remain fixed. In this section, a linear approximation to the dynamical model similar to that proposed in [302] is considered, allowing efficient proposals. If no linear approximation is available, what follows could be adapted to work with the extended or unscented Kalman filters. Alternatively, the Particle Gibbs method of the next section can be used.

Such 'bridging' proposal distributions can be sampled in a manner similar to backward sampling from a smoothing distribution since, using the conditional independence structure of the model,

$$p(X_{t_1:t_2}|X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2}) = \prod_{t=t_1}^{t_2} p(X_t|X_{t+1}, X_{t_1-1}, y_{t_1:t}),$$

and

$$p(X_t|X_{t+1}, X_{t_1-1}, y_{t_1:t}) = \frac{p(X_{t+1} \mid X_t)p(X_t \mid X_{t_1-1}, y_{t_1:t})}{p(X_{t+1} \mid X_{t_1-1}, y_{t_1:t})}.$$

The distribution $p(X_t \mid X_{t_1-1}, y_{t_1:t})$ is the filter distribution for X_t with initial state X_{t_1-1}.

The approximation is formed by using a linear Gaussian approximation of the transition density $q(X_{t+1} \mid X_t) \sim \mathcal{N}(A_t X_{t+1}, Q_t)$, where A_t is the approximating state transition matrix at time t and Q_t is the approximating covariance of the state transition at t. For example, to approximate the motion of a single object using the near constant velocity model these should be set as

$$A_t = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \qquad Q_t = \begin{bmatrix} h^3/3 & h^2/2 \\ h^2/2 & h \end{bmatrix},$$

for each object (giving a block-diagonal structure for multiple objects), where h is the inter-observation time-step.

For such linear Gaussian models the forward-filter distribution of the state X_t (with initial state X_{t_1-1}) is given by the Kalman filter and is denoted $q(X_t \mid X_{t_1-1}, y_{t_1:t}) \sim \mathcal{N}\left(\mu_{t|t}, \Sigma_{t|t}\right)$. Identities (A.1) and (A.2) from appendix A can be used to obtain a distribution for X_t conditional on the subsequent state X_{t+1} as

$$q(X_t | X_{t+1}, X_{t_1-1}, y_{t_1:t}) \sim \mathcal{N}\left(\mu_{t|t_2+1}, \Sigma_{t|t_2+1}\right) \tag{6.29}$$

with

$$\begin{aligned} \Sigma_{t|t_2+1} &= \left(A_t' Q_t A_t + \Sigma_{t|t}^{-1}\right)^{-1} \\ \mu_{t|t_2+1} &= \Sigma_{t|t_2+1}\left(A_t' Q_t^{-1} X_{t+1} + \Sigma_{t|t}^{-1}\mu_{t|t}\right). \end{aligned}$$

This can be used to propose a state sequence by successive sampling from equation (6.29), starting with X_{t_2} and working back to X_{t_1}.

Such proposals do not depend on the current samples for $X_{t_1:t_2}$, giving an independence sampler with acceptance probability

$$p_{\text{accept}} = \min\left(1, \frac{p(X^*_{t_1:t_2} | X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2}) q(X^{\text{cur}}_{t_1:t_2} | X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2})}{p(X^{\text{cur}}_{t_1:t_2} | X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2}) q(X^*_{t_1:t_2} | X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2})}\right). \tag{6.30}$$

Since

$$p(X_{t_1:t_2}|X_{t_1-1}, X_{t_2+1}, y_{t_1:t_2}) \propto \prod_{t=t_1}^{t_2+1} p(y_t \mid X_t)p(X_t \mid X_{t-1}),$$

and similarly for q, the acceptance ratio can be evaluated straightforwardly. When the observation model is linear and so can be shared between the proposal and target, i.e. when $q(y_t \mid X_t) = p(y_t \mid X_t)$, these cancel in the acceptance ratio, giving the simpler form

$$p_{\text{accept}} = \min\left(1, \frac{\prod_{t=t_1}^{t_2+1} p(X_t^* \mid X_{t-1}^*)q(X_t^{\text{cur}} \mid X_{t-1}^{\text{cur}})}{\prod_{t=t_1}^{t_2+1} p(X_t^{\text{cur}} \mid X_{t-1}^{\text{cur}})q(X_t^* \mid X_{t-1}^*)}\right),$$

where $X_{t_1-1}^* = X_{t_1-1}^{\text{cur}}$ and $X_{t_2+1}^* = X_{t_2+1}^{\text{cur}}$.

Sampling all object paths simultaneously might lead to low acceptance rates if there are many objects. A subset of object paths can be sampled at a time by making proposals in which new states are proposed only for a subset of objects, with that of the other objects being left unchanged. This allows pathwise (i.e. one object at a time) sampling, albeit requiring evaluation of the full transition density. This proposal strategy is useful for weakly coupled or independent objects.

6.6.3 State Inference via Particle Gibbs

The Particle Gibbs method of [33], described in section 4.3.4, is an alternative method for state inference that allows new sample paths to be generated from an approximate particle filter algorithm targeting the required posterior. These paths are (up to the approximation introduced by numerical integration) exact samples from the posterior. This allows the use of fully non-linear and non-Gaussian models in sample path generation, and therefore offers a method more likely to succeed in cases where model non-linearities make proposing from bridging distributions too challenging.

For full-state filtering and backward sampling (i.e. sampling the state of all objects simultaneously), the filter and smoother algorithms in sections

4.3.4 and 4.3.5 can be directly applied, using the full state transition and observation densities discussed in section 6.6.1. However, as in the previous section, it is possible to use Particle Gibbs to make pathwise per-object (or per-object subset) samples from the posterior, which is especially useful with large numbers of relatively weakly coupled objects, where the standard algorithm will almost certainly fall foul of the curse of dimensionality. This is discussed in the following section.

Particle Gibbs sampling also permits construction of a forward simulation-only algorithm avoiding the evaluation of transition densities, although this does not permit sampling of subsets of objects or the use of backward sampling, since both of these require the evaluation of transition densities. Such simulation-only algorithms are therefore likely to suffer from poor mixing under all but the mildest circumstances with few objects and amenable models. On the other hand, they offer a potentially powerful way to deal with intractable transition models, and, in some circumstances when combined with exact diffusion sampling techniques [59; 60] (themselves only applicable to a limited subset of diffusions), offer a theoretical way of building MCMC schemes targeting the exact posteriors of models with intractable densities. Slow mixing in such methods could perhaps be improved in some cases by simulating from bridging distributions (of the transition model only) if these can be calculated, as suggested in [33], although that has not been explored further here.

Pathwise Sampling with Particle Gibbs

Pathwise Particle Gibbs sampling can be achieved, at the expense of evaluating the full state transition density, by applying the Particle Gibbs algorithm of section 4.3.4 to a sub-block of the state trajectory, in which the sub-block is the trajectory of a single target (or group of targets). This alternative blocking scheme is suggested in the authors' reply to the comments in [33] (the main text in [33] suggests blocking schemes in time, with blocks containing the trajectory of all targets). This 'pathwise' scheme ne-

cessitates the use of a conditional particle filter that is conditioned on the full trajectories of all objects other than the one being sampled, and this is described below (in this section) for the problem being considered here.

The PGibbs algorithm in section 4.3.4 is modified as follows, with bold text highlighting changes from the PGibbs algorithm in section 4.3.4.

- Sample $\theta \sim p(\theta \mid X_{0:T}, y_{1:T})$.
- Sample $X_{0:T}^{\text{sample}} \sim p(X_{0:T} \mid \theta, y_{1:T})$ via the steps:

  For each object i:

  Sample $X_{\mathbf{i},0:T}^{-k}, a_{1:T}^{-k} \sim \tilde{\pi}(X_{\mathbf{i},0:T}^{-k}, a_{1:T}^{-k} \mid k, \mathbf{X_{-i,0:T}^{-k}}, x_{0:T}^{k}, a_{1:T}^{k}, \theta)$,

  Sample $k \sim \tilde{\pi}(k \mid X_{0:T}, a_{1:T}, \theta)$,

  $X_{0:T}^{\text{sample}} = X_{0:T}^{k}$ is a sample from $p(X_{0:T} \mid \theta, y_{1:T})$.

The key change here is that instead of sampling the whole of $X_{0:T}^{-k}$ from the extended target conditional $\tilde{\pi}(X_{0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{0:T}^{k}, a_{1:T}^{k}, \theta)$, only the $X_{i,0:T}^{-k}$ pertaining to object i are sampled from their extended target conditional $\tilde{\pi}(X_{i,0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{-i,0:T}^{-k}, X_{0:T}^{k}, a_{1:T}^{k}, \theta)$. This is simply a blocking scheme for the Gibbs sampler (see section 2.2), with the state variables for an object i forming the block of variables to be sampled (other than those from the path selected by k, i.e. the ancestry of particle k at time T, henceforth referred to as the k^{th} path; see section 4.3.4).

From the definition of the conditional target $\tilde{\pi}(X_{0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{0:T}^{k}, a_{1:T}^{k}, \theta)$ in equation (4.18),

$$\tilde{\pi}(X_{0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{0:T}^{k}, a_{1:T}^{k}, \theta) = \psi(X_{0:T}^{-b_{0:T}}, a_{1:T}^{-b_{1:T}} \mid X_{0:T}^{b_{0:T}}, b_{0:T})$$

where b_t is the index of the particle on the k^{th} path at time t (with $X_{0:T}^{-k} \equiv X_{0:T}^{-b_{0:T}}$ and $a_{1:T}^{-k} \equiv a_{1:T}^{-b_{1:T}}$). The required conditional of the target distribution for pathwise Gibbs sampling is given by

$$\begin{aligned}\tilde{\pi}(X_{i,0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{-i,0:T}^{-k}, X_{0:T}^{k}, a_{1:T}^{k}, \theta) &= \tilde{\pi}(X_{i,0:T}^{-k}, a_{1:T}^{-k} \mid k, X_{-i,0:T}^{-k}, X_{0:T}^{k}, a_{1:T}^{k}, \theta) \\ &= \psi(X_{i,0:T}^{-b_{0:T}}, a_{1:T}^{-b_{1:T}} \mid X_{0:T}^{b_{0:T}}, X_{-i,0:T}^{-k}, b_{0:T})\end{aligned}$$

where ψ is the distribution of the variables generated in the particle filter, defined in equation (4.9), $X_{i,0:T}^{-k}$ denotes the sampled state of the i^{th} object at all times across all particles other than those selected by k (i.e. b_t at time t). The distribution $\psi(X_{i,0:T}^{-b_{0:T}}, a_{1:T}^{-b_{1:T}} \mid X_{0:T}^{b_{0:T}}, X_{-i,0:T}^{-k}, a_{1:T}^{b_{1:T}})$ is the conditional distribution of the variables generated in the particle filter, conditioned on both the selected k^{th} path variables ($X_{0:T}^{b_{0:T}}$ and $b_{0:T}$), and the non-object i state variables $X_{-i,0:T}^{-k}$.

Sampling from this distribution can be achieved by running a conditional particle filter in which only new values of $X_{i,0:T}^{j}$ and $a_{1:T}^{j}$ are sampled (conditioned on the other variables, which remain unchanged) for all particles other than that on the k^{th} path (i.e. $j = b_t$ at time t). Initial particles should be drawn from the conditional prior, i.e. for initial particles $j = 1, ..., b_0 - 1, b_0 + 1, ..., N_0$, draw

$$\begin{aligned} X_{i,0}^{j} &\sim p(X_{i,0}^{j} \mid X_{-i,0}^{j}), \\ v_0^{j} &= 1/N_0. \end{aligned}$$

(An importance distribution q_0 could also be sampled and the weights adjusted appropriately). At subsequent time steps, the variable $X_{i,t+1}^{j}$ is sampled from $q(X_{i,t+1}^{j} \mid X_{-i,t+1}^{j}, X_t^{a_{t+1}^{j}}, y_{1:t+1})$, and the variable a_{t+1}^{j} is sampled from $R(a_{t+1}^{j} \mid v_t)$ for particles $j = 1, ..., b_t - 1, b_t + 1, ..., N_0$; $X_{i,t}^{b_t}$ and $a_{t+1}^{b_{t+1}} = b_t$ remain unchanged)

The particle weights v_t are calculated as a function of the sampled variables (including those not pertaining to object i and thus not being resampled on this pass), as in the particle filter algorithm in section 4.3.1. This entails calculating the un-normalized weight w_t^{j}, followed by normalization to ensure the v_t^{j} sum to 1. Given the sampled variables, w_{t+1}^{j} is calculated as follows (as in equation (4.7))

$$w_{t+1}^{i} = \frac{p(y_{t+1} \mid X_{t+1}^{j}) p(X_{t+1}^{j} \mid X_t^{a_{t+1}^{j}}) v_t^{j}}{q(X_{t+1}^{j} \mid X_t^{a_{t+1}^{j}}, y_{1:t+1}) R(a_{t+1}^{j} \mid v_t)},$$

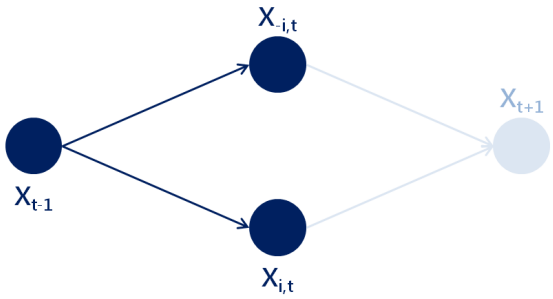


Figure 6.17: One step ahead transition structure for discrete or single step integrated systems. Future states (pale blue) depend on the full system state at t, but at t the state of objects i and that of the other objects is conditional independent given the previous state

where here $X^j_{t+1} = X^j_{i,t+1} \cup X^j_{-i,t+1}$, with v^j_{t+1} being found as

$$v^i_{t+1} = \frac{w^i_{t+1}}{\sum_{i=1}^{N_{t+1}} w^i_{t+1}}.$$

Unfortunately in the general case re-calculation of w^i_{t+1} requires the evaluation of the state transition and proposal densities across *all* objects, not just object i, which can be computationally costly. In some special cases known conditional independence relationships between objects could be exploited to speed up computation.

In cases when no intermediate state variables are introduced between observation times (e.g. for discrete models or continuous models when single step integration is used, but unlike multi-step integration using Bayesian imputations), a minor simplification is possible. In that case

$$\begin{aligned} p(X^j_{t+1} \mid X_t^{a^j_{t+1}}) &= p(X^j_{i,t+1}, X^j_{-i,t+1} \mid X_t^{a^j_{t+1}}, X^j_{-i,t}) \\ &= p(X^j_{i,t+1} \mid X_{i,t}^{a^j_{t+1}}, X^j_{-i,t}) p(X^j_{-i,t+1} \mid x_{i,t}^{a^j_{t+1}}, X^j_{-i,t}), \end{aligned}$$

where the second line is possible because the state of object i at time t is conditionally independent of the state of the other objects at time t, given the previous state, providing no future states are conditioned on (see figure 6.17). Under these conditions, a convenient form of the filter (somewhat

akin to the standard bootstrap particle filter) is given by choosing

$$q(X^j_{i,t+1} \mid X^j_{-i,t+1}, X_t^{a^j_{t+1}}, y_{1:t+1}) = p(X^j_{i,t+1} \mid X_{i,t}^{a^j_{t+1}}, X_{-i,t}),$$

i.e. sampling new states $X^j_{i,t+1}$ from the transition density for object i, which if used alongside the multinomial resampling scheme $R(a^j_{t+1} \mid v_t) = v^j_t$, gives the un-normalized weights as

$$w^j_{t+1} = p(y_{t+1} \mid X^j_{i,t+1}, X^j_{-i,t+1}) p(X^j_{-i,t+1} \mid X_{i,t}^{a^j_{t+1}}, X_{-i,t}).$$

Here the expensive calculation of $p(X_{-i,t+1} \mid x_{i,t}^{a^j_{t+1}}, X_{-i,t})$ need only be calculated for each *ancestor* that is chosen, so a degenerate filter with only a few ancestors has the consolation of quicker weight calculation. This form of pathwise Particle Gibbs sampling is used in section 6.7. In general, the calculation of the full transition density (or almost full transition density in the above special case) for the entire state will be the most computationally expensive part of the sampling process.

The subsequent step to sample k can be completed as in the standard Particle Gibbs sampler (see section 4.3.4) by, in the simplest case, selecting a particle k with probability v^j_T to give a new k^{th} path. Thus it is possible to perform conditional sampling of individual object paths within the Particle Gibbs framework, and the conditional samples drawn will be exact (aside from integration error in cases where exact integration of the underlying dynamics is not possible).

As noted in section 4.3.5, mixing in Particle Gibbs methods can be improved using backward sampling. For pathwise sampling as described here, the method set out in that section can be followed. The state transition densities that must be calculated are the full state transition densities, so that the density $p(x_{t+1}^{b_{t+1}} \mid x_t^j)$ appearing in equation (4.20) when calculating the backward-sampling distribution of the sample trajectory indices $b_{0:T}$ is given by the full state transition density for all objects from t to $t+1$. This is necessary because the state of $X_{i,t}$ could affect all other objects at $t+1$.

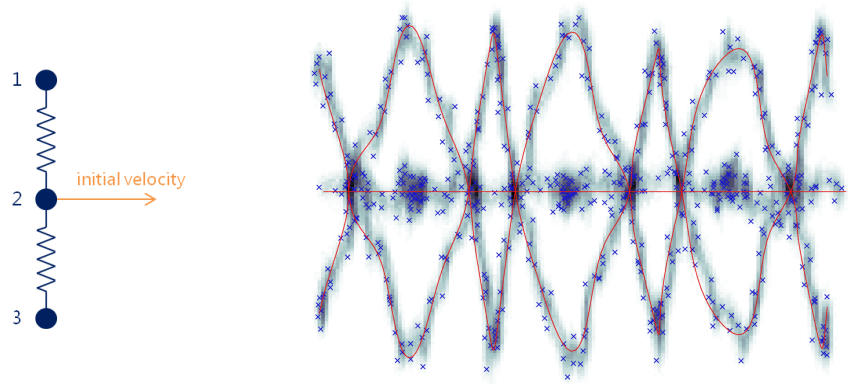


Figure 6.18: Three object setup (left) and resulting paths (red, right), with observations shown as blue crosses. Shading represents object positions inferred in simultaneous linkage and position inference, with strength of shading indicating frequency of object paths passing through each pixel

The only exception to this would be if objects could be divided into independent groups, in which case each group could be treated separately, both in backward sampling and in forward filtering, saving substantial time.

6.7 Results with Noisy Observations

This section shows the results of simultaneous object path and linkage estimation. The first section briefly examines the different proposed path inference approaches, though this is by no means an exhaustive test, more an assessment of which method is most appropriate to the example problems being tackled. The second section shows some examples of linkage learning applied to noisy data and shows successful structure inference for an 8 object system.

6.7.1 Path Estimation Methods

In order to compare the various path estimation methods they were tested on the noisy paths of three objects, connected in a rope configuration with linear springs. The central object was given an initial velocity. Figure 6.18 shows the object configuration and the resulting object tracks (the in-

ferred path density shown is taken from an algorithm in which linkage was learned). The system was simulated with no state transition noise and observations are given by the object positions distorted with additive Gaussian noise. To remove one variable between tests, all methods were supplied with the correct linkage functions and their gradients at the bin centres, and these were not estimated during the run. The tests were run on a series of 200 observations, with observation noise variance $\sigma^2_{\text{obs}} = 0.01$. This corresponds roughly to the noise level shown in figure 6.20. All runs were initialized to the observation positions perturbed by additive Gaussian noise with the same variance as the observation noise.

Figure 6.19 shows the error and mixing results for seven path inference methods: site-by-site Gibbs sampling, block and pathwise Gibbs sampling with bridge proposals, and block and pathwise Particle Gibbs sampling with and without back sampling. Tests results are shown with respect to computation time, since the methods take significantly different amounts of time to sample each site. For example, in the test in figure 6.19, the Gibbs sampler sampled each site about 9,000 times per hour, whereas the pathwise Particle Gibbs sampler with block sampling sampled each site about 12 times per hour). The methods were all implemented in Matlab and shared code (for example transition density evaluation) where possible. The error results show the mean absolute error per object state per time period. Mixing results show the sum of absolute differences between the current path sample and that 1000s previously, giving some indication of the amount of variation in the samples over time. Combined with low error, high mixing levels are desirable because they indicate that the method is better able to explore the space of possible solutions, rather than become trapped at locally good solutions. The "RTS baseline" in figure 6.19 refers to a baseline estimation produced by taking independent samples from the linear smoothing distribution using an independent constant velocity model for all objects.

Both site-by-site and bridge-proposal based Gibbs sampling arrived at

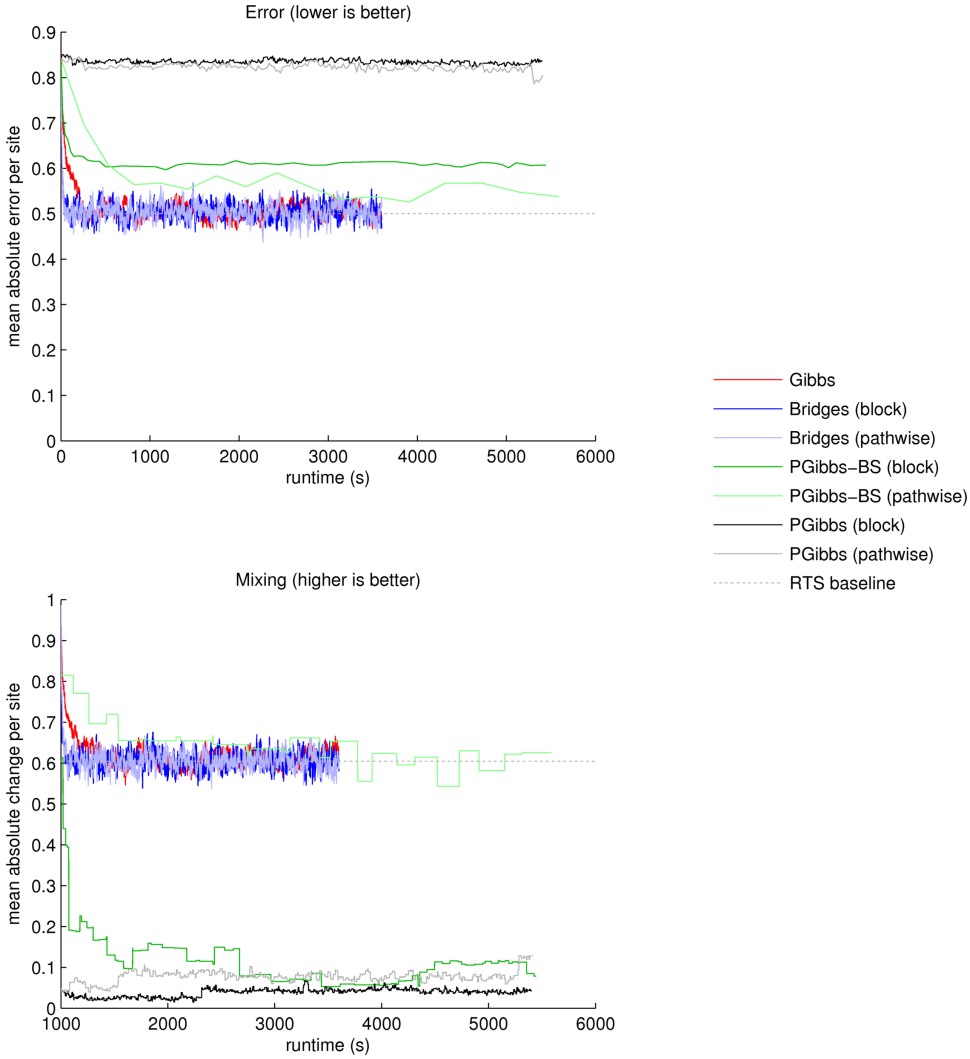


Figure 6.19: Mean absolute error and mixing for a range of state inference techniques using fixed state transition noise σ_p^2=0.5. Mixing is taken to be the sum of absolute differences between samples at 1000s lag

low error solutions quickly and mixed well. The initial convergence of the bridge-based samplers was substantially better than that of the site-by-site sampler, due to the local strong correlation of the object states. For the site-by-site Gibbs sampler, acceptance rates for each type of proposal were in the following ranges: random walk 0-6%, predictive 4-12%, adapted 4-12%. In the test, a mixture of all three proposal types was used. Bridge-based proposals achieved acceptance rates of 5%-30% for block proposals and 10%-50% for pathwise proposals. The proposed bridges were uniformly random lengths at uniformly random locations up to one-eighth of the length of the time series.

The performance of Particle Gibbs without backward sampling, both with block and pathwise proposals, was poor, barely reducing error from the initial estimate. This is due to the low levels of mixing, particularly in early parts of the time series, that plague this method; almost no new particles (proposals) are accepted. These results support the idea that spending additional time on backward sampling is worthwhile. In this case, with 80 particles, backward sampling takes roughly ten times as long per sample as forward filtering alone. With backward sampling (PGibbs-BS in figure 6.19), Particle Gibbs methods are, perhaps surprisingly, almost competitive with respect to computational effort with the standard Gibbs sampling methods. In particular, the pathwise method, though slow, achieves reasonable initial convergence, and error and mixing levels in similar ranges to those for the Gibbs samplers. On the other hand, when also estimating the process noise variance σ_p^2 (not shown here), these methods do not perform so well simply because of their very low rates of sampling, which do not allow other system variables such as σ_p^2 the opportunity to converge.

For Particle Gibbs methods in particular, pathwise sampling proved to be more effective in both reducing error and improving mixing than block sampling, even considering its higher computational cost. Since this test was conducted with only three objects, this is a strong result, as increasing the number of objects will almost certainly increase the advantage of

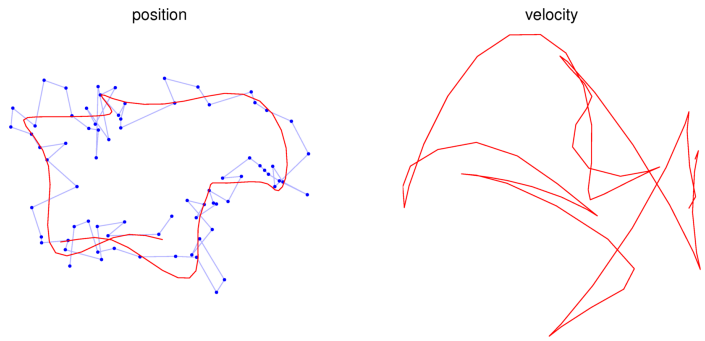


Figure 6.20: Example noisy observations (position only) with $\sigma^2_{obs} = 0.005$ of object 1 in the example setup in figure 6.5 for the first 75 timesteps of 0.2s, comparable to those in figure 6.7 . The blue dots show observations (linked by pale blue lines); red line shows the true object state

pathwise sampling, except when the objects are tightly coupled.

With these results as a guide, a mixture of site-by-site Gibbs sampling and pathwise bridge based Gibbs sampling was used in the tests that follow. This mixture allowed rapid initial convergence and showed good mixing properties whilst being quick to run and thus allowing more samples to be drawn of other variables including linkage estimates. The problems tackled thus far do not deviate sufficiently from the linear case to warrant the additional complexity of Particle Gibbs methods, but this test should not be taken as a dismissal of those methods; they are likely to perform better in severely nonlinear situations.

6.7.2 Linkage Inference with Noisy Observations

In order to test the inference of object linkage with noisy observations, a test similar to those that produced figures 6.14 and 6.15 (second panel) was run using data from the setup shown in figure 6.5. In this case only observations of object positions were supplied to the algorithm and these were distorted with additive Gaussian noise with variance $\sigma^2_{obs} = 0.005$, as illustrated in figure 6.20. A mixture of site-by-site and bridge-based Gibbs sampling was used for position inference. The linkage inference results are shown in figure 6.21. The results show an underestimation of the strength

of the linkage functions and greater uncertainty in their values, compared to those in figure 6.14 and the lower panel of figure 6.15, although the rough shape of the linkage functions was correctly inferred. A similar effect was seen when noisy data was pre-smoothed using a linear smoother before processing in the non-noisy algorithm, so the effect could be related to excessive path smoothing. However, a longer series of observations allows accurate, low variance inference to be made of the linkage, as shown in figure 6.22. It is unsurprising that the data requirements are higher with noisy observations, since the noise causes the loss of information in those observations. The runtime for these two examples was about 100 minutes for that with 200 observations and about 6.5 hours for that with 800 observations (about 1.2s and 4.8s per sample, respectively), showing linear time scaling with $|T|$.

Figure 6.24 shows the linkage inferred through application of the algorithm to a dataset of 600 observations derived from eight moving objects linked as shown in figure 6.23. Observations were subject to additive Gaussian noise of variance $\sigma^2_{obs} = 0.005$. These results show clear inference of the system structure, albeit with slight underestimation of the spring strengths, particularly towards the edge of the function domains. This could indicate an insufficient number of observations as in the five object case and is probably exacerbated by the edge effects suffered by Gaussian processes, as mentioned in section 6.5.3. The runtime for this test was about 14 hours, corresponding to about 10s per sample.

As with non-path inference, the estimates of the functions themselves are much better than those of the gradients. This is to be expected, but in particular the variance of the estimates might give cause for concern with respect to the numerical integration scheme in appendix D, which was used here. Since this relies on the Jacobian, high variance in the gradient samples could lead to poor integration performance. As the gradients are sampled and path inference is conditioned on these samples, very high variance estimates could even be detrimental to performance, especially

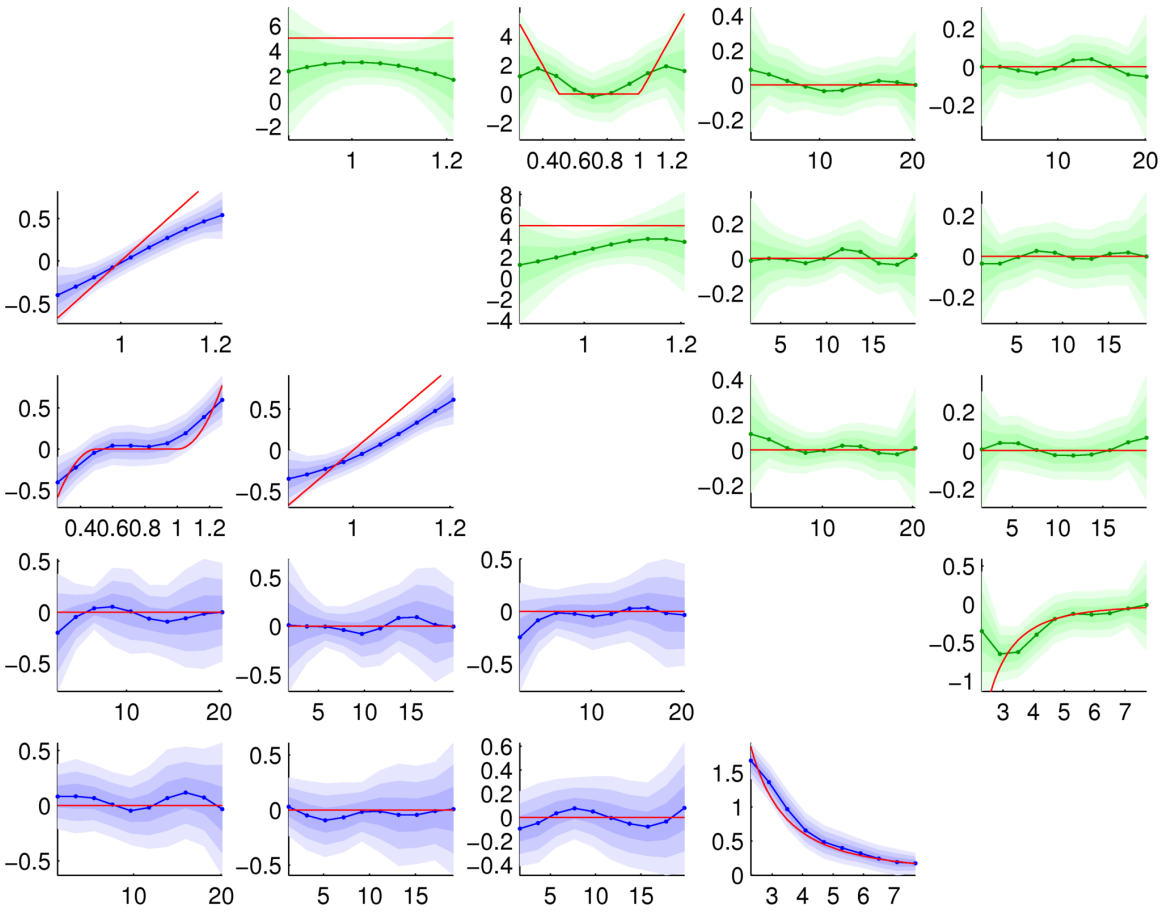


Figure 6.21: Linkage (lower left) and linkage gradient (upper right) inference with noisy observations; setup is as in figure 6.5, using 200 object position observations with 0.2s timestep and observation noise variance $\sigma^2_{obs} = 0.005$. In all graphs x-axis is inter-object distance and y-axis is linkage strength f_{ij} (blue, lower-left graphs) or linkage gradient f'_{ij} (green, upper-right graphs); red lines indicate true values. Results from 3000 samples, after a 2000 sample burn-in

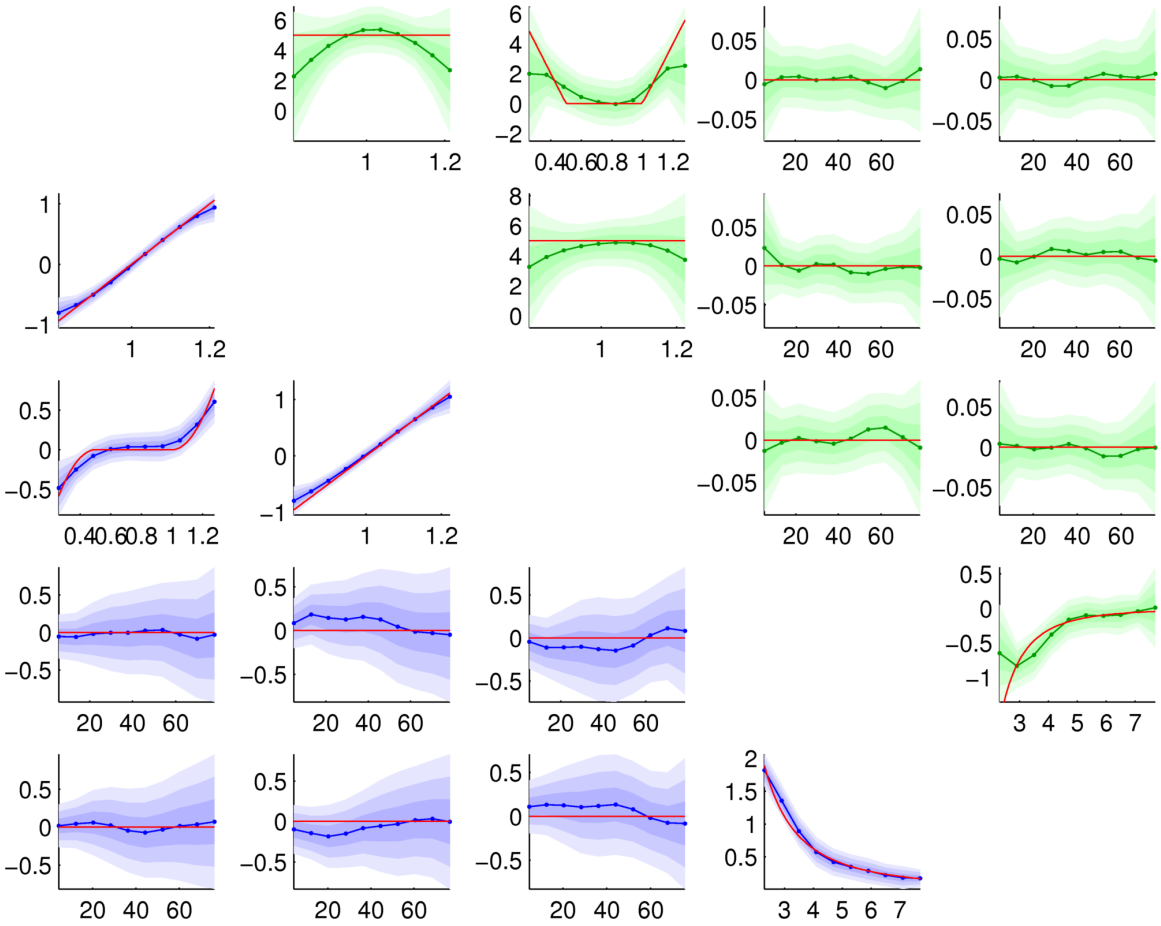


Figure 6.22: Linkage and linkage gradient inference with noisy observations; setup is as in figure 6.5, using 800 object position observations with 0.2s timestep and observation noise variance $\sigma^2_{obs} = 0.005$. In all graphs x-axis is inter-object distance and y-axis is linkage strength f_{ij} (blue, lower-left graphs) or linkage gradient f'_{ij} (green, upper-right graphs); red lines indicate true values. Results from 3000 samples following a 2000 sample burn-in

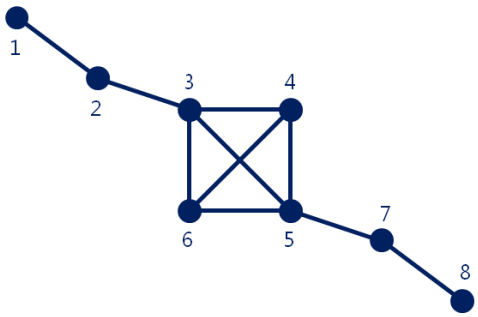


Figure 6.23: Object connections in eight object test data. Lines indicate linear springs of natural length 1, other than those between objects 3 and 5, and 4 and 6, which have natural length $\sqrt{2}$

in algorithms such as the Particle Gibbs with bootstrap proposals, where large magnitude Jacobians could result in very high variance proposals and possibly numerical instability.

In the examples shown, the effect of linkage inference on state inference performance is mixed. In all the tests run, the state inference error of the two methods was found to be within about 10% of each other, with the best performing algorithm different on different runs. This variation is lower than that arising from using different sets of observations with the same noise characteristics. However, knowledge of linkage greatly improves one-step prediction accuracy. Correct knowledge of linkage reduced the RMS prediction error to about $1/2$, $1/3$ and $1/6$ of that without linkage information for the three, five and eight datasets, respectively. Since prediction is an important component of tracking, it could be expected that the inference of linkage would improve tracking performance, though this is yet to be confirmed via experiment.

6.8 Conclusion

The algorithm presented in this chapter offers a new way of determining the nature of a useful class of relationships between interacting objects. Specifically, the algorithm can be applied in cases where inter-object rela-

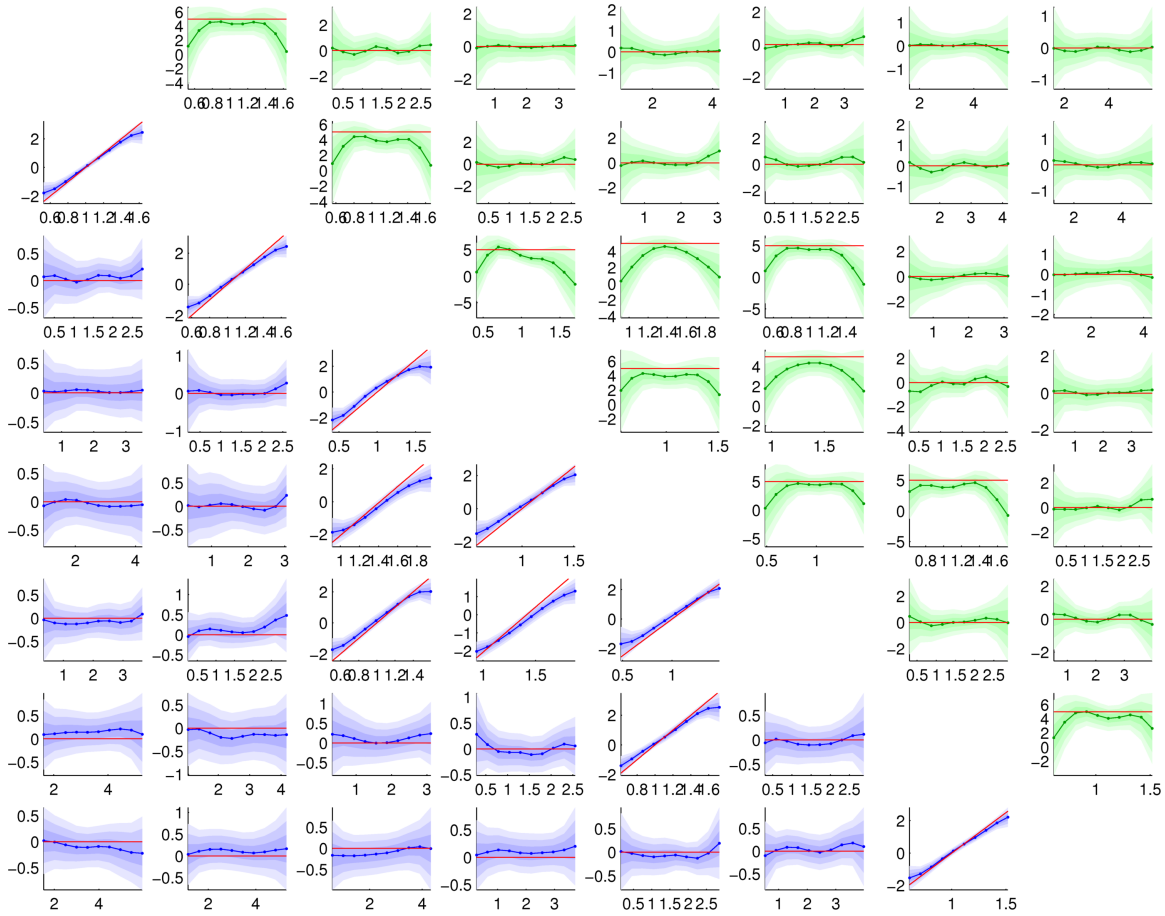


Figure 6.24: Linkage and linkage gradient inference with noisy observations; setup is as in figure 6.23, using 600 object position observations with 0.2s timestep and observation noise variance $\sigma^2_{obs} = 0.005$. In all graphs x-axis is inter-object distance and y-axis is linkage strength f_{ij} (blue, lower-left graphs) or linkage gradient f'_{ij} (green, upper-right graphs); red lines indicate true values. Results from 3000 samples following a 2000 sample burn-in

tionships are functions of some one-dimensional quantity of the joint object state, in a 'direction' (in state space) that can be derived from the joint object states. This class contains a large number of useful interactions, such as many distance or velocity based interactions in systems of physical objects. It could also be applied to other time series models such as those arising in econometrics or biology, but the exposition in this chapter has focussed on physical object interaction, due to its possible applicability to tracking applications and its intuitive interpretation. The algorithm described makes only non-parametric assumptions about the shape of the inter-object relationships and, as has been demonstrated, due to this is able to identify a range of relationship types without prior assumptions and without user intervention other than easily interpretable length scale parameters for the processes, which could themselves be estimated.

The original version of this method, presented in [4], was limited by computational complexity that scaled with the cube of the number of observations, making it intractable for large problems. That problem has been overcome in this work through the use of a binning strategy to sparsify the Gaussian process inference. Approximate error results are available for this approach that suggest it is a consistent estimator. The bin-based approach is highly effective in practice, producing results comparable to those from the dense method, and with computational complexity scaling linearly with the number of observations. This makes the method plausible for long time series, greatly increasing its usefulness. Access to long time series is also crucial for estimation with many objects, since in this case a very large number of relationships must be estimated, requiring a great deal of information to be available in the form of observations. Problems probably arising from insufficient data were seen in section 6.7.

A further problem with the method in [4] was a limited ability to cope with noisy observations. This work has attempted to overcome this by the addition of a state inference layer to the original system that is able to incorporate inferred linkage information. This has been fairly successful

and has allowed noisy and incomplete (e.g. position only) observations to be used for state inference. However, no particular improvement in position inference over methods not considering object linkage was found, and this might suggest that, in the cases examined, much simpler independent smoothing for each object would have produced equally good results. This seems somewhat surprising given that linkage is clearly detectable in the series examined. On the other hand, knowledge of linkage greatly improved state prediction performance, and so correct linkage inference can be expected to improve tracking performance.

Several methods were developed for state inference, including a version of the Particle Gibbs algorithm using pathwise inference. Whilst this did not offer a particular advantage over Gibbs sampling methods for the problems tackled, it did show a significant advantage in terms of mixing over Particle Gibbs sampling all targets simultaneously, which can suffer from the curse of dimensionality. The method is computationally expensive, however, requiring a very similar amount of computation to the standard method (which jointly samples all objects) for every object. Under certain circumstances (highly nonlinear model, not too strong linkage between objects) this method is likely to be the most suitable of all proposed methods; developing an example of such a situation remains a subject for future work.

The method presented has several limitations compared to existing methods for structure inference. It is slow compared to (approximate) linear methods such as [47] and, unlike the linear models used in tracking in [293], [270] and [294] is not yet in a sequential form, although this appears to be technical feasible and is a key objective of future research.

Non-parametric, non-linear continuous DBN approaches for gene regulatory networks, such as those of [286], [285] and [289] are the existing approaches most closely related to the work presented here. This work can be seen as an extension of these methods in several ways, as well as their application to a different domain. Firstly, the objects considered

here cannot be represented with a single variable but rather each have multi-dimensional state as shown in figure 6.3, with the internal dynamical model of each object (i.e. that between the variables of which the object is composed) assumed known, but interaction between object having to be learnt. Secondly, the models used here deal with a continuous time problem, rather than the discrete time one usually considered in gene regulatory network inference. This continuous-time approach is more appropriate to physical problems and allows the incorporation of asynchronous observations if necessary. Thirdly, a different regression model for the linkage functions based on Gaussian process regression is developed, which allows a computationally efficient and fully Bayesian inference algorithm to be developed. Finally, noisy observations, potentially from a non-linear, non-Gaussian observation model, can be used to infer network structure.

A possible limitation of the specific continuous time scheme used here is the use of an integration scheme requiring the evaluation of Jacobian matrices. As seen in the results, gradient inference is considerably less accurate and estimates suffer much higher variance than those of the linkage functions themselves. The effect of these poor estimates on integration performance has not been systematically examined, but it is possible that multi-step Euler-Maruyama schemes (see appendix D) could be preferable in some cases, in spite of their substantially higher computational cost. Such schemes would affect the derivation of inter-object force and this would have to be reformulated accordingly.

The methods presented here provide a useful tool for the analysis of an important class of networks of interacting objects and could find applications in several domains including tracking, computer vision, biology, finance and econometrics. They extend existing non-linear non-parametric DBN methods in useful ways. Though not currently in a sequential form, this work points the way to the development of non-linear group interaction models for tracking applications.

6.8.1 Further Work

There are a number of extensions and developments that could further enhance the algorithms presented here. Perhaps the most interesting extension would be to devise a sequential version of the algorithm. This would allow the inference of non-linear linkage in tracking applications for the first time. It might even be possible to Rao-Blackwellize part of the linkage inference to create an efficient scheme, as in the model in chapter 5.

An obvious extension to the model presented is to allow interaction between objects based on relative velocity. This would require a different numerical integration scheme than that in appendix D, although that scheme could easily be adapted to account for this. The inclusion of velocity relationships would allows motion such as flocking behaviour to be investigated with these techniques.

As discussed in section 6.5, Gaussian process regression is equivalent to a certain type of kernel regression. If the Gaussian process schemes proposed are deemed too slow, other types of kernel regression schemes such as local linear regression could be used directly as is done in e.g. [286], [285] and [289]. This might make the formulation of a Bayesian solution more challenging (although [289] does something similar for spline models). Local linear regression in particular is known to exhibit much reduced edge effects in comparison to Watson-Nadaraya estimators [299] and so might have an advantage in this respect over the Gaussian process regression used here, where these effects are clearly visible, especially in gradient estimation.

The force inference scheme described in section 6.2 (and also used in chapter 5) actually wastes information, since the force term also appears in the expression for the change in velocity, albeit with higher noise, which could relatively easily be incorporated into this method as an additional pseudo-observation. It would be interesting to see if this makes a significant contribution to the accuracy of inference and to its data requirements.

The results in this chapter do not test two aspects already incorpor-

ated into the model. Firstly, network sparsity estimation, using indicator functions, has had some success in preliminary trials and could lead to large efficiency improvements if disjoint groups of objects can be automatically identified. These could then be treated separately in, for example, transition density calculation, potentially reducing the $O(N^2)$ complexity of the method to $O(kn^2)$ where $n < N$. Secondly, only symmetric relationships between objects have been examined, though this is not a limitation of the method as presented. The removal of this assumption would allow the method to make one-directional causal inference about the relationships between objects if such relationships are present. Combined with sparsity estimation, this would allow a succinct estimate of causal structure between the objects to be produced.

Finally, a good test of the method would be to compare predictions of the model to those of algorithms making linear assumptions, with and without the presence of non-linear effects.

Chapter 7

Sparse Audio Restoration

This chapter presents a method for audio noise reduction in the case when the original signal is corrupted by both homogenous background noise and impulse noise. It extends the background noise removal algorithm of [303] to the case where impulse noise is also present by introducing a sparse impulse process with variable scale. This is similar to the earlier work in [1], although there it was necessary to sample an intermediate z process representing the true signal distorted with homogenous Gaussian in order to apply the background noise removal method of [303]; here it is shown how to marginalize out that intermediate process from the conditional distributions necessary for sampling. Inference is carried out by means of a Gibbs sampling scheme for all variables.

Background noise is a common feature of many audio tracks and arises from a number of sources such as thermal noise arising in recording or processing equipment. As such, it is present, usually at the same scale, throughout the track. Impulse noise, on the other hand, takes the form of large but brief deviations between the observed value and the true signal. Impulses can be caused by, amongst other things, wear, dirt and scratches on vinyl records, and are perceived as audible pops and clicks in a recording. Because it can derive from multiple sources and, in the case of vinyl recordings, involves uncontrolled deviation of the playback needle, im-

pulse noise can vary across a very wide range of scales. Much previous impulse removal work has been carried out using autoregressive methods, described in [283], for example, though these have a smoothing effect on the signal, acting as a form of low-pass filter and causing the loss of some high-frequency detail.

Early work in noise reduction can be found in [304], but the area continues to be active, e.g. [305; 306]. An overview of a range of methods can be found in [283] and the references therein, but alternative psychoacoustically-based approaches such as [232] have also been popular. A technique common to several methods, and used here for background noise removal, is the representation of the signal as a weighted sum of basis functions, with the aim being to reconstruct the true signal without reconstructing the noise. Since the composition of audio signals varies with time, decomposition is performed in blocks on short sub-sections of the whole signal and, in order to reduce blocking effects, these sub-sections generally overlap; see figure 7.1. The localized functions of varying frequencies used in such reconstructions are often called *wavelets* and a collection of such wavelets covering the full time span of the signal forms a *dictionary* of basis functions into which the original signal can be decomposed. There are many possible choices of wavelet dictionaries with various properties; common choices include modified discrete cosines functions [307], which provide an orthogonal basis, and Gabor functions, used in this work, [303; 308], which do not.

The presence of overlapping non-orthogonal wavelets in the dictionary leads to multiple possible decompositions of the signal into the (local) basis functions, known as *over-completeness*. Of these, sparse representations are frequently preferred as they give a parsimonious representation of the original signal. Numerous methods of finding good sparse signal representation exist [309; 310; 303], with the choice of which of these is 'best' dependent on the application. For example, a representation that minimizes the number of non-zero coefficients might be best for compression, whilst

one that has stronger temporal structure between components is likely to be better for missing data reconstruction [308]. The approach taken in [303] and followed here focuses on explicitly modelling sparsity through the use of indicator variables ($\in \{0, 1\}$) that determine whether or not a particular basis function is included in the signal representation. This formulation allows the straightforward incorporation of prior models of signal structure, meaning that expectations about likely sparsity structure such as certain types of temporal coherence can be embedded within the prior. This model-based approach is conceptually distinct from approaches that determine coefficients in such a way as to target sparsity directly, almost all of which attempt to limit or penalize the L_1 norm of the regression coefficients (i.e. the sum of the coefficient magnitudes) such as those of [310; 309; 311]. In a Bayesian setting a similar result can be achieved through the use of a Laplacian prior on the basis coefficients, producing a 'penalty' term on the L_1 norm in the posterior.

In this work, structural priors are also applied to the modelling of impulse noise. Impulses are assumed to either be present or absent in each digital audio sample, as defined by an indicator i_t. Priors on these indicators can then be used to incorporate expectations about the impulse's temporal structure. Section 7.3 shows how a two state Markov chain, for which the parameters can be estimated, can be used to express an expectation that impulses will be rare, but are likely to last for several samples when they do occur. Given this prior structure, the indicators can be sampled using a Gibbs sampler as shown in section 7.4.

The rest of this chapter is structured as follows. Section 7.1 introduces the Gabor wavelet decomposition used as the basis of background noise reduction. Section 7.2 explains the models of background and impulse noise used for noise reduction. Section 7.3 gives details of the structured sparsity priors used for impulse removal. Section 7.4 outlines the Gibbs samplers used for inference, with the necessary conditional distributions derived there. Section 7.5 shows how the intermediate z process used in

[1] can be marginalized out of the inference process and thus need not be sampled. Section 7.6 presents results of noise removal, showing that the approach outlined here works with both artificial and real noise, and finally section 7.7 draws conclusions and suggests further work.

7.1 Gabor Signal Decomposition

In [303] the authors use *Gabor signal decomposition* as the basis for background noise reduction. This decomposition is the process of taking a signal and representing it as a weighted sum of *Gabor synthesis atoms* localized in time and frequency. A signal of length L can be decomposed into $M \times N$ Gabor synthesis atoms, representing M discrete frequency levels and N discrete time points, arranged as a grid. Such a transform maps a continuous signal $x(t)$ onto an $M \times N$ time-frequency plane as shown in figure 7.1.

The Gabor synthesis atoms are defined in general by

$$\tilde{g}_{m,n}(t) = g\left(t - \frac{n}{N}L\right) \exp\left(2\pi i \frac{m}{M} t\right), \tag{7.1}$$

where $m \in \{0, 1, ..., M-1\}$, $n \in \{0, 1, ..., N-1\}$ and, for discrete samples as in digital audio, $t \in \{0, 1, ..., L-1\}$. Figure 7.2 shows some examples of Gabor synthesis atoms. The function g in equation 7.1 is the *Gabor window function*, typically a smooth bell-shaped window function with compact support that defines the temporal envelope of the corresponding Gabor atoms. The method here uses a Hann window, defined as

$$g(t) = \begin{cases} 0.5 + 0.5\cos\left(2\pi t/\lambda\right) & |t| \leq \lambda/2 \\ 0 & |t| > \lambda/2 \end{cases}, \tag{7.2}$$

where λ defines the window width, but many other choices are possible, including the Bartlett, Blackman, (truncated) Gaussian, Hamming, Kaiser, and Tukey windows, each centred at the parameter value and having slightly different shapes and characteristics; the choice of window functions is dis-

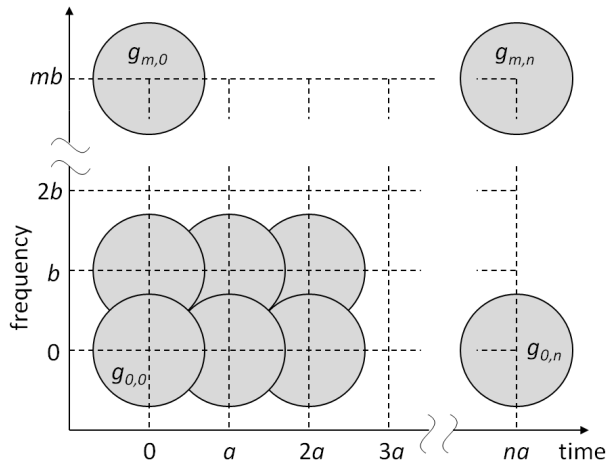


Figure 7.1: A lapped transform, formed of overlapping atoms $g_{m,n}$ arranged in a regular grid in time-frequency space. These atoms form the basis for the representation of the signal in time-frequency space

cussed further in [312]. The width of the chosen window function must be such that it provides sufficient overlap between synthesis atoms (i.e. somewhat larger than L/N).

Given a set of synthesis atoms $\tilde{g}_{m,n}(t)$, a (complex) input signal $x(t)$ can be written as their weighted sum:

$$x(t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \gamma_{m,n} c_{m,n} \tilde{g}_{m,n}(t), \tag{7.3}$$

where $c_{m,n} \in \mathbb{C}$ is the weighting coefficient for each atom and $\gamma_{m,n} \in \{0, 1\}$ are indicator variables that determine whether a particular atom is present in the decomposition. These are key to imposing sparse structure within the model, discussed further in section 7.3. The Gabor representation can be written in matrix-vector form as $x = \tilde{G}\tilde{c}$, where the input signal is represented as a column vector $x = [x(0)\ x(1)\ ...\ x(L-1)]^T$, $\tilde{G}$ is the $L \times MN$ *Gabor synthesis matrix*, consisting of the $(m, n)^{th}$ Gabor synthesis atom at each signal observation time as its $(m + nM)^{th}$ column, and the coefficient vector $\tilde{c}$ is formed by stacking the individual coefficients (multiplied by the

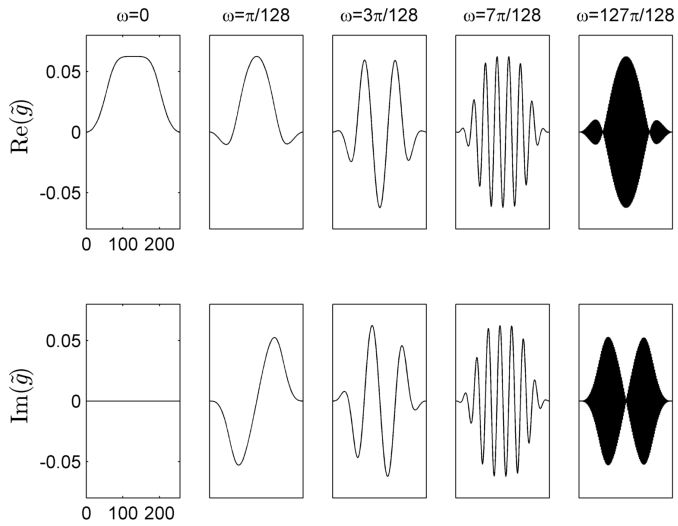


Figure 7.2: A selection of Gabor synthesis atoms (real and complex parts) generated using a Hann window of width 256 (modified to generate basis functions forming a tight frame) with frequencies ω

corresponding indicator) $\gamma_{m,n}c_{m,n}$ in the appropriate order (see figure 7.3).

For decompositions in which the number of Gabor synthesis atoms is greater than the number of observations ($MN > L$), the system $x = \tilde{G}\tilde{c}$ is under-determined with respect to the coefficients c. This is the case in almost all real applications since redundancy in the Gabor dictionary is necessary in order to achieve good time-frequency localization. This is a consequence of the Balian-Low theorem [313; 314], which states that there is no well-concentrated Gabor basis in the critically sampled case where $MN = L$, discussed in more detail in [303] and [312]. The under-determined system $x = \tilde{G}\tilde{c}$ can be solved via the *Gabor transform*, in which the coefficient of each atom is found by taking the inner product of that atom with the signal. Because atoms have compact support this can be performed efficiently using only the part of the signal that corresponds to the atom's region of support; [315] gives an algorithm for the discrete Gabor transform. Though this has the property that it recovers the coefficients that are minimal in an L_2 sense (i.e. they have minimal sum-of-squares), there is no guarantee of sparsity of coefficients. Indeed, this is unlikely in

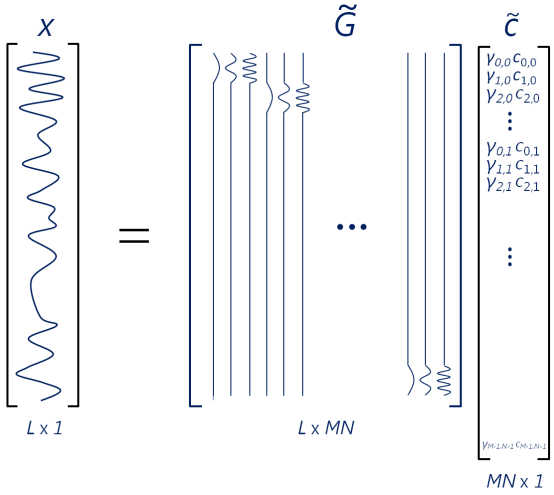


Figure 7.3: Signal decomposition using a set of synthesis atoms can be thought of as regression aiming to reconstruct the signal x from the synthesis atoms. In matrix-vector form, each synthesis atom forms one column of the $\tilde{G}$ matrix. In the Gabor case, each atom has compact support and is a shifted version of the corresponding atom at the previous time location

general as the L_2 norm will penalize the use of a few large coefficients as opposed to a larger number of smaller ones.

If the input signal is entirely real, as is the case with the audio signals considered here, the expansion on the right hand side of equation (7.3) must also be real. Assuming that M is even, this can be arranged by setting $c_{m,n} = c^*_{M-m,n}$ for all $m \in \{1, 2, ..., M/2\}$, relying on the fact that $\tilde{g}_{m,n} = \tilde{g}^*_{M-m,n}$, which can readily be shown from the definition of the Gabor synthesis atoms in equation (7.1). In this case the decomposition in equation (7.3) can be written as

$$\begin{aligned} x(t) &= \sum_{m=0}^{M/2} \sum_{n=0}^{N-1} \gamma_{m,n} \alpha_m (c_{m,n} \tilde{g}_{m,n}(t) + c^*_{m,n} \tilde{g}^*_{m,n}(t)) \\ &= \sum_{m=0}^{M/2} \sum_{n=0}^{N-1} \gamma_{m,n} \left(\Re(\alpha_m c_{m,n}) \Re(\tilde{g}_{m,n}(t)) - \Im(\alpha_m c_{m,n}) \Im(\tilde{g}_{m,n}(t)) \right), \end{aligned} \quad (7.4)$$

where α_m is 1 for all m except for $m = 0$ and $m = M/2$, when it is $1/2$. This allows the decomposition to be reformulated in matrix-vector form using

only real numbers by redefining $\tilde{G}$ and c to be entirely real, containing interleaved real and (negated) imaginary components as shown in appendix A.1 of [303]. Given these definitions, $\tilde{G}\tilde{c}$ will be the signal reconstruction in equation (7.4). For practical purposes in what follows, the $c'_{m,n}$ coefficients will be treated as a two element vector of real numbers, representing the real and imaginary components of $c'_{m,n}$. This will be denoted $c_k \in \mathbb{R}^2$, with $k \in \{0, 1, ..., (M/2+1)N-1\}$ so that $c_k = c'_{m+nM}$ corresponds to $c'_{m,n}$.

7.2 Signal Model

The audio signal model in [303] assumes that the received audio samples are composed of the true signal at the sample time, corrupted by homogenous additive Gaussian noise. At each sample time $t = 0, ..., L-1$ the received signal y_t is composed of the true signal $x(t)$ distorted by additive Gaussian noise ν_t so that

$$y_t = x(t) + \nu_t,$$

with $\nu_t \sim \mathcal{N}(0, \sigma^2_{\nu_t})$. Homogenous background noise as in [303] is modelled by having a constant noise scale across all samples, so that $\sigma_{\nu_t} = \sigma$ throughout, where σ is a parameter of the model that can be estimated. This can be extended to model for the possible presence of impulse noise in the received signal by allowing the scale of the noise process to increase when such impulse noise is present. The noise scale is then given by

$$\sigma^2_{\nu_t} = (1 + i_t\lambda_t)\sigma^2. \tag{7.5}$$

where $i_t \in \{0, 1\}$ is an indicator variable determining whether impulse noise is present at a particular sample time t, and λ_t gives a scale for the impulse at that time if it exists. Thus the noise variance is σ^2 when no impulse is present and $(1 + \lambda_t)\sigma^2$ when it is.

A simple choice for λ_t is to set it to be constant, say λ_{fixed}. However,

since impulsive noise can originate from a number of different physical sources, a single scale factor λ_{fixed} might not lead to a noise distribution sufficiently heavy-tailed to capture all impulses. Therefore the scale factor λ_t can be allowed to vary with time, giving an impulse scale at each sample time which can be estimated.

Although in principle many prior structures $p(\lambda_t)$ are possible for λ_t, a convenient one, as used in [306] in a different context, is a shifted inverse gamma model, the shape of which is shown in figure 7.4. This is a truncated and shifted version of the inverse gamma distribution (note the offset of +1 in the λ_t arguments) and takes the form

$$\begin{aligned} p(\lambda_t) &= \frac{\beta_\lambda^{\alpha_\lambda}(1+\lambda_t)^{-(\alpha_\lambda+1)}\exp(-\beta_\lambda/(1+\lambda_t))}{\gamma(\alpha_\lambda,\beta_\lambda)}, \quad \lambda \geq 0, \\ &\propto \mathcal{IG}\left(1+\lambda_t;\alpha_\lambda,\beta_\lambda\right) \end{aligned} \tag{7.6}$$

where $\mathcal{IG}\left(1+\lambda_t;\alpha_\lambda,\beta_\lambda\right)$ is the inverse gamma pdf with parameters α_λ and β_λ, evaluated at $1+\lambda_t$ and $\gamma(\alpha_\lambda,\beta_\lambda)$ is the lower incomplete gamma function defined as

$$\gamma(\alpha_\lambda,\beta_\lambda) = \int_0^\beta t^{\alpha-1}e^{-t}dt. \tag{7.7}$$

Since the Gabor-based noise reduction mechanism described in [303] is based on the assumption of homogenous background noise it cannot be applied directly to input signals containing impulse noise of the type described. This can be overcome by introducing an artificial latent process z with the required homogenous noise distribution such that

$$z_t = x(t) + w_t. \tag{7.8}$$

with $w_t \sim \mathcal{N}(0,\sigma^2)$, which allows the original Gabor decomposition algorithm to be applied to this process. The z process can be inferred by

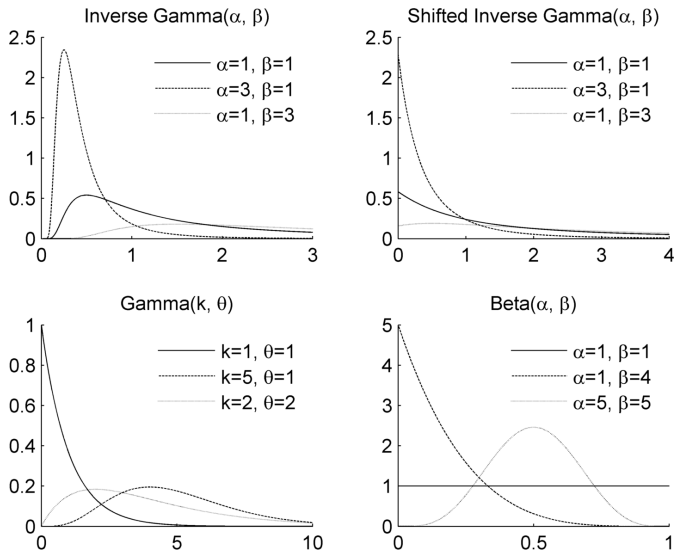


Figure 7.4: Probability density functions for a number of priors used in the model with a selection of parameter values

applying the impulse removal mechanism to the observations y_t since

$$y_t = z_t + i_t u_t, \tag{7.9}$$

with $u_t \sim \mathcal{N}(0, \lambda_t \sigma^2)$. This structure is shown in figure 7.5 and has the property that the true signal x (i.e. the values of $x(t)$ at the sample times, denoted x_t for discrete sample times) is conditionally independent of the observations y and impulse indicators i, given the z process, so that

$$p(x \mid y, z, i, \lambda) \quad \propto p(x \mid z),$$

where here un-subscripted variables have been used to indicate the full set of such variables (e.g. $i = \{i_t \mid t \in 0, ..., L-1\}$). This means that samples from the posterior distribution $p(x \mid z)$ can be drawn as in [303], by applying the algorithm there to a sample of the latent process z rather than directly to the input samples y. In this scheme, a sampling iteration consists of sampling both the z and x process along with the other model variables and parameters, although section 7.5 shows how explicit sampling of the z

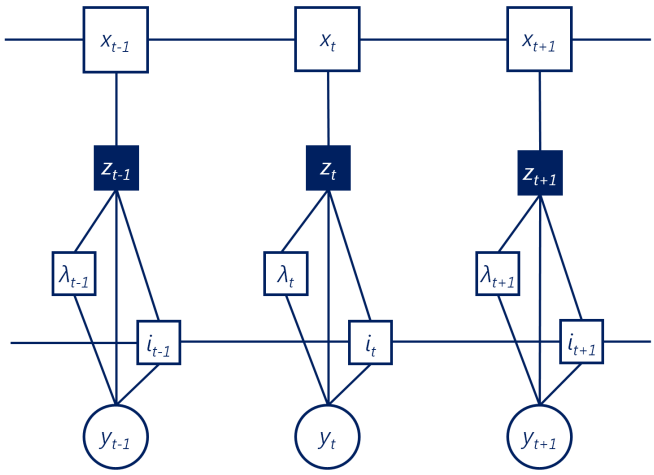


Figure 7.5: Logical structure of model variables showing the artificial latent z process along with impulse indicators i and scale factors λ. The z process is the true signal distorted by homogenous Gaussian noise whereas observations y may be subject to noise at multiple scales

process can be avoided.

7.3 Structured Sparsity

Prior distributions for the indicator variables (for both impulses and Gabor coefficients) are important components of the model. It is through these priors that a preference for sparsity can be incorporated, since they can encode a belief that sparse solutions are more likely than dense ones. Unlike methods that specifically seek a minimal solution in some norm, Bayesian inference does not inherently favour any particular solution unless that solution is more probable according to the modelling and prior assumptions. The over-completeness of the Gabor dictionary and the flexibility that this introduces means that without some sort of regularization there is a strong risk of over-fitting the Gabor coefficients to the noisy signal; the modelling and prior assumptions are what prevent this.

The priors on the sets of indicator variables $\gamma = \{\gamma_{m,n} \mid \forall m, n\}$ and $i = \{i_t \mid t = 0, ..., L-1\}$ can be used to encode a prior belief that solutions

will be sparse in terms of Gabor coefficients and impulses. In many cases, however, further prior information about the structure of the non-zero indicators is available and it is desirable to incorporate this in the model via the indicator priors, leading to the idea of structured sparsity.

Consider the impulse process represented by the i variables, indicating the presence or absence of an impulse at a particular sample time. It is likely that impulses will be present in relatively few samples (i will be sparse) and this simple expectation can be incorporated into the prior in a straightforward way, through a prior belief that an indicator value of 0 (no impulse) is more likely than 1 (impulse present). A more sophisticated prior model can incorporate the belief that impulses will be relatively rare but, when they do occur, are likely to last for a number of samples, since the time taken to traverse a damaged section of record surface is likely to be longer than a single sample. In this case, the prior encodes a belief about the likely structure of the i process.

The simplest prior for i is to treat each i_t as a Bernoulli random variable with some prior probability p of a sample being subject to an impulse. This alone is sufficient to favour sparse solutions, since if p is small, a sparse solution is, all other things being equal, more likely than a dense one. Under these assumptions, the prior probability p indicates the proportion of samples that might be expected to be affected by impulse noise. The prior on the full set of indicators i in this case is given by

$$p(i \mid \phi_i) = \prod_{t=0}^{L-1} p(i_t \mid \phi_i),$$

where ϕ_i is the set of parameters for the prior on i. In the Bernoulli case this is just the prior probability $p \in \phi_i$ of an indicator being 1, so that

$$\begin{aligned} p(i_t = 1) &= p, \\ p(i_t = 0) &= 1 - p. \end{aligned}$$

Here a link can be made to penalized likelihood estimation, a common alternative method for finding sparse solutions. In such methods the sparse estimator is one that maximizes a version of the log-likelihood function penalized according to the number of non-zero coefficients, with the strength of the penalty being determined a penalty coefficient η, chosen by the user. For the impulse indicator variable this can be expressed as

$$\hat{i}_{\text{PLE}} = \arg\max_{i} \log p(y \mid i) - \eta \|i\|_0. \tag{7.10}$$

where $\|i\|_0$ is the number of non-zero elements of i.

The Bayesian posterior distribution of the indicator variables i given the observations is

$$\log p(i \mid y) = \log p(y|i) + \log p(i) + C,$$

where C is constant with respect to i. For the Bernoulli prior above, this becomes

$$\log p(i \mid y) = \log p(y|i) + \log\left(\frac{p}{1-p}\right)\|i\|_0 + C',$$

and thus the penalized likelihood estimate in equation (7.10) is equivalent to a *maximum a posteriori* (MAP) estimate from the Bayesian model (that is, the estimate that maximizes the posterior density) with the Bernoulli prior, where $\eta = \log(p/1-p)$. When $p < 0.5$ this is negative, resulting in a penalty term for additional non-zero coefficients. It is perhaps more intuitive to choose a prior probability p in the Bayesian formulation than it is to choose a value for the penalty coefficient η in the penalized likelihood approach.

The Bayesian formulation allows further complexity to be built into the prior assumption in a simple and explicit way. In order to incorporate a belief that impulses, when they do occur, are likely to last for several samples, the prior for the impulse indicator can be modelled as a two-state Markov chain. The idea behind this is that in the 'no impulse' state, the next state

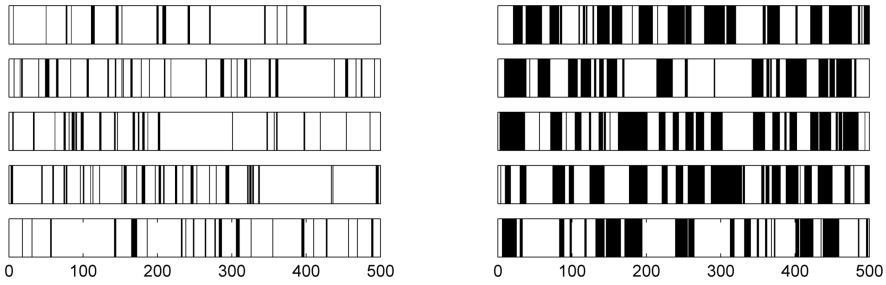


Figure 7.6: Sample draws of length 500 from the Markov chain prior with $p_{00} = 0.95$, $p_{11} = 0.5$ for the left group of five draws and $p_{00} = 0.9$, $p_{11} = 0.9$ for the right group (black indicates a value of 1)

of the indicator process is very likely to also be 'no impulse', with only a small probability of a transition to the 'impulse' state. However, once in the 'impulse' state, the next state is quite likely also to be an 'impulse', with some probability of a transition back to 'no impulse'. Figure 7.6 shows some draws from such a Markov chain prior with different transition probabilities. In this case

$$p(i \mid \phi_i) = p(i_0 \mid \phi_i) \prod_{t=1}^{L-1} p(i_t | i_{t-1}, \phi_i),$$

and the conditional distribution of a particular indicator i_t given the rest of the indicator process is given by

$$p(i_t \mid i_{-t}, \phi_i) \propto p(i_{t+1} \mid i_t, \phi_i) p(i_t \mid i_{t-1}, \phi_i),$$

where $p(i_{t+1} \mid i_t, \phi_i)$ is determined by the transition probabilities of the Markov chain (the notation i_{-t} refers to the set of all i indicators, excluding that at sample time t, i.e. $i_{-t} = i \backslash i_t$). The transition probabilities can be taken to be parameters of the model or can themselves be inferred from the data, as detailed in section 7.4. Two parameters define the Markov chain transition matrix: p_{00}, the probability of remaining in state 0, and p_{11}, the probability of remaining in state 1 (the other entries in the transition matrix can be calculated from these).

In general the inference methods in section 7.4 can use any conditional prior $p(i_t \mid i_{-t}, \phi)$ for the indicator variables. This is a very flexible class of possible prior functions and means that many different forms of prior knowledge can be incorporated in this framework. Incorporating more structure in the prior can lead to less sparse results, since structural priors impose additional restrictions on the solution compared to simple Bernoulli priors. On the other hand, structured priors can give lead to better results if the structure in the prior is a good model of the true process.

Similar prior structures can also be used for the Gabor coefficient indicators γ, as described in [303]. Simple Bernoulli priors giving a prior probability for each atom being zero lead to sparse solutions in time-frequency space, though possibly with little structure between atoms, especially if the prior probability of a non-zero coefficient is small. This might be most suitable for compression, where minimizing the number of non-zero coefficients is paramount. As with the impulse process i, Markov chain priors can be imposed in time, implying that frequency components have some tendency to remain consistent from one sample block to the next. Such a prior structure might be appropriate for signals expected to consist of slowly time-varying oscillations and, as with the impulse indicator prior, the transition probabilities can be estimated from the data as shown in section 7.4. Similarly, a Markov chain structure can be imposed in the frequency direction, implying a prior expectation of local frequency clustering in each of the N sample blocks. Another possible prior for the Gabor coefficients is a Markov random field (MRF) prior, which can be used to impose two dimensional structure on the coefficients. Such priors favour signals in which activity occurs in patches on the time-frequency plane.

7.4 Inference

The joint posterior distribution of the Gabor reconstruction variables (c, γ and others used in the model in [303]), latent process variables (z), im-

pulse process variables (i, ϕ_i and λ) and noise scale parameter (σ) can be sampled using a Gibbs sampler. The variables involved in estimating the Gabor reconstruction of the signal, c, γ, σ^2 and several others (σ_c^2, ν and ϕ_γ, corresponding to prior parameters for the Gabor coefficients c and their indicators γ), can each be sampled using the conditional distributions given in [303], replacing the observations on which those distributions are conditioned with the intermediate z process defined in equation 7.8, since this process has the same noise characteristics as the observations used in [303].

The variables corresponding to the impulse process at a given sample time, i_t, z_t and λ_t, can be sampled as a block from their joint conditional distribution

$$\begin{aligned} p(i_t, z_t, \lambda_t \mid x, y, i_{-t}, z_{-t}, \lambda_{-t}, \sigma^2, \phi_i) = \\ p(z_t \mid i_t, \lambda_t, x_t, y_t, \sigma^2) p(\lambda_t \mid i_t, x_t, y_t, \sigma^2) p(i_t \mid i_{-t}, x_t, y_t, \sigma^2, \phi_i), \end{aligned} \tag{7.11}$$

where x denotes the signal reconstruction from the Gabor synthesis atoms as in equation (7.3), with x_t denoting its value at the time of input sample t. A joint sample can be drawn by by sampling i_t, λ_t and z_t sequentially (in that order) from the distributions on the right of equation (7.11).

The distribution from which to sample i_t is given by

$$p(i_t \mid i_{-t}, x_t, y_t, \sigma^2, \phi_i) \propto p(i_t \mid i_{-t}, \phi_i) p(y_t \mid x_t, i_t, \sigma^2). \tag{7.12}$$

The impulse indicator i_t is a Bernoulli random variable and can be sampled by evaluating the ratio r_t of the expression in equation (7.12) for both possible values of i_t, so that

$$r_t = \frac{p(i_t = 1 \mid i_{-t}, \phi_i) p(y_t \mid x_t, i_t = 1, \sigma^2)}{p(i_t = 0 \mid i_{-t}, \phi_i) p(y_t \mid x_t, i_t = 0, \sigma^2)}. \tag{7.13}$$

The posterior probability of $i_t = 1$ is then given by

$$p(i_t = 1 \mid i_{-t}, x_t, y_t, \sigma^2, \phi_i) = \frac{r_t}{1 + r_t},$$

so that $i_t \sim \text{Bernoulli}(\frac{r_t}{1+r_t})$, which can easily be sampled.

In the simple case where $\lambda_t = \lambda_{\text{fixed}}$ for all t, the observation likelihood is given by

$$p(y_t \mid x_t, i_t, \sigma^2) \quad = \mathcal{N}\left(y_t \mid x_t, (1 + i_t\lambda_{\text{fixed}})\sigma^2\right),$$

and for non-constant impulse noise scale, the likelihood is given by

$$p(y_t \mid x_t, i_t, \sigma^2) = \begin{cases} \mathcal{N}\left(y_t \mid x_t, \sigma^2\right), & i_t = 0 \\ p(y_t \mid x_t, i_t = 1, \sigma^2), & i_t = 1 \end{cases}. \tag{7.14}$$

If the prior $p(\lambda_t)$ is an inverse gamma distribution of the form in equation (7.6), $p(y_t \mid x_t, i_t = 1, \sigma^2)$ can be found in closed form, as described in [306]:

$$\begin{aligned} p(y_t \mid x_t, i_t = 1, \sigma^2) &= \int_0^\infty p(y_t \mid \lambda_t, x_t, i_t = 1, \sigma^2) p(\lambda_t) d\lambda_t \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \frac{\gamma(\alpha_p, \beta_p)}{\gamma(\alpha_\lambda, \beta_\lambda)} \frac{\beta_\lambda^{\alpha_\lambda}}{\beta_p^{\alpha_p}}, \end{aligned} \tag{7.15}$$

where

$$\begin{aligned} \alpha_p &= \alpha_\lambda + 1/2 \\ \beta_p &= \beta_\lambda + \frac{(y_t - x_t)^2}{2\sigma^2}, \end{aligned}$$

and where $\gamma(\alpha_p, \beta_p)$ is defined as in equation (7.7). Thus all quantities necessary for calculation of the ratio r_t in equation (7.13) can be evaluated by evaluating the prior $p(i_t \mid i_{-t}, \phi_i)$ and likelihood $p(y_t \mid x_t, i_t, \sigma^2)$ for the cases $i_t = 1$ and $i_t = 0$.

With a sample drawn for i_t, λ_t can be drawn from the conditional dis-

tribution

$$\begin{aligned} p(\lambda_t \mid i_t, x_t, y_t, \sigma^2) &\propto p(y_t \mid x_t, \lambda_t, i_t, \sigma^2) p(\lambda_t) \\ &= \mathcal{N}\left(y_t \mid x_t, (1 + i_t\lambda_t)\sigma^2\right) p(\lambda_t) \\ &\propto \begin{cases} p(\lambda_t), & i_t = 0 \\ \mathcal{IG}\left(1 + \lambda_t; \alpha_p, \beta_p\right), & i_t = 1. \end{cases} \end{aligned} \tag{7.16}$$

Assuming an inverse gamma prior $p(\lambda_t)$ for λ_t as in equation (7.6), both distributions in the last line of equation (7.16) are shifted inverse gamma distributions and can be sampled using a rejection sampling trick. First, a variable $l_t = 1 + \lambda_t$ is defined. This is then sampled from the inverse gamma distribution with the appropriate parameters. If the sampled value is less than 1, it is rejected and the variable resampled, otherwise it is accepted and 1 is subtracted from it to give a sample for λ_t. This can be shown to result in a sample from the required distribution. Since $1 + \lambda_t$ is distributed according to an inverse gamma distribution, the noise variance $\sigma^2_{\nu_t} = \sigma^2(1 + i_t\lambda_t)$ is scaled by an inverse gamma random variable when an impulse is present ($i_t = 1$). In this case, the noise is drawn from a scale mixture of Gaussians, which can be shown to have a Student t-distribution using the result [316; 317] that

$$t_{2\alpha}(x \mid \mu, \sigma^2) = \int_0^\infty \mathcal{N}\left(x; \mu, \sigma^2 s\right) \mathcal{IG}\left(s; \alpha, \alpha\right) ds,$$

i.e. that if $X \sim \mathcal{N}\left(\mu, \sigma^2 S\right)$ with $S \sim \mathcal{IG}(\alpha, \alpha)$, then $X \sim t_{2\alpha}(\mu, \sigma^2)$. By noting that $\frac{\beta}{\alpha} S \sim \mathcal{IG}\left(\alpha, \beta\right)$, then scaling X by $\sqrt{\frac{\beta}{\alpha}}$, it can be seen that for $X \sim \mathcal{N}\left(\mu, \sigma^2 R\right)$, $R \sim \mathcal{IG}(\alpha, \beta)$, then $X \sim t_{2\alpha}\left(\mu, \frac{\beta}{\alpha}\sigma^2\right)$. Thus, when an impulse is present, the noise is distributed $\nu_t \sim t_{2\alpha_p}\left(0, \frac{\beta_p}{\alpha_p}\sigma^2\right)$.

Finally, once i_t and λ_t have been sampled, z_t can be sampled from the

conditional distribution

$$\begin{aligned} p(z_t \mid i_t, \lambda_t, x, y) &\propto p(y_t \mid z_t, i_t) p(z_t \mid x_t) \\ &= \mathcal{N}(y_t \mid z_t, i_t \lambda_t \sigma^2) \mathcal{N}(z_t \mid x_t, \sigma^2) \\ &\propto \mathcal{N}\left(z_t \mid \frac{y_t + i_t \lambda_t x_t}{1 + i_t \lambda_t}, \frac{i_t \lambda_t \sigma^2}{1 + i_t \lambda_t}\right). \end{aligned} \tag{7.17}$$

Note that if $i_t = 0$ then $z_t = y_t$.

The parameters of the indicator prior ϕ_i depend on the prior structure chosen for the impulse indicators i. In general, the distribution of the parameter(s) is given by

$$p(\phi_i \mid i) \propto p(i \mid \phi_i) p(\phi_i) \tag{7.18}$$

$$= p(\phi_i) \prod_t p(i_t \mid i_{\mathcal{N}(t)}, \phi_i), \tag{7.19}$$

where $\mathcal{N}(t)$ is the neighbourhood of indicators that influence the prior of the indicator i_t. For the Bernoulli and Markov chain prior structures outlined in section 7.3 the prior parameters are given as follows.

- *Bernoulli*: In the Bernoulli case, the neighbourhood of each i_t is empty ($\mathcal{N}(t) = \emptyset$) for all t and the 'likelihood' term (the product in equation (7.19)) is given simply by $p^{|i|}(1-p)^{L-|i|}$, where $|i|$ is the number of non-zero elements of i. The conjugate prior for this Bernoulli likelihood is the beta distribution $\mathcal{B}(\alpha_i, \beta_i)$, which, for $\alpha_i = \beta_i = 1$, gives a uniform prior (see figure 7.4). Using this prior it is possible to marginalize out the Bernoulli parameter p when calculating the ratio $\frac{p(i_t=1|i_{-t})}{p(i_t=0|i_{-t})}$ in the expression for r_t in equation (7.13) (see [303], appendix A.3). In this case, that ratio is given by

  $$\frac{p(i_t = 1 \mid i_{-t})}{p(i_t = 0 \mid i_{-t})} = \frac{|i_{-t}| + \alpha_i}{L - |i_{-t}| - 1 + \beta_i},$$

  where $|i_{-t}|$ is the number of non-zero indicators excluding i_t.

- *Markov chain*: The transition matrix for the Markov chain prior is fully

determined by the probability of remaining in state 0, p_{00}, and the probability of remaining in state 1, p_{11} (since $p_{01} = 1 - p_{00}$ and similarly for p_{10}). These can be estimated in the same way as the parameters of Markov chain priors for Gabor coefficient indicators γ in [303] (appendix A.4) in which the transition probabilities are treated as independent Bernoulli variables with beta prior distributions. The initial distribution of the chain $p(i_0 \mid p_{00})$ is taken to be the chain's stationary distribution. Then, for example for p_{00},

$$\begin{aligned} p(p_{00} \mid i) &\propto p(i \mid p_{00}) p(p_{00}) && (7.20) \\ &\propto p(p_{00}) p(i_0 \mid p_{00}) \prod_{t \in \{t \mid i_{t-1}=0\}} p(i_t \mid i_{t-1}, p_{00}). && (7.21) \end{aligned}$$

Since the transition probabilities from state 0 are considered as independent of those from state 1, only indicators whose predecessor is 0 need be considered. A similar expression can be derived for p_{11}.

Sampling can be performed using a Metropolis-within-Gibbs step. This is particularly convenient if a beta prior $\mathcal{B}(\alpha_{p_{00}}, \beta_{p_{00}})$ is applied to p_{00} and proposals p_{00}^* are drawn from the full conditional distribution in equation (7.21) where the initial state instead has a fixed, uniform distribution (i.e. $p(i_0 \mid p_{00}) = p$), leading to a tractable proposal distribution defined as

$$\begin{aligned} q(p_{00}^* \mid p_{00}^{(i)}) &= p(p_{00}^*) \prod_{t \in \{t \mid i_{t-1}=0\}} p(i_t \mid i_{t-1}, p_{00}^*) && (7.22) \\ &= \mathcal{B}\left(|A_{00}| + \alpha_{p_{00}}, |A_{01}| + \beta_{p_{00}}\right), && (7.23) \end{aligned}$$

where $p_{00}^{(i)}$ is the current sample of p_{00}, and A_{00} is the set of times of transitions from 0 to 0, i.e.

$$\begin{aligned} A_{00} &= \{t \mid i_{t-1} = 0, i_t = 0\}, \\ A_{01} &= \{t \mid i_{t-1} = 0, i_t = 1\}. \end{aligned}$$

Thus $|\mathcal{A}_{00}|$ is the number of transitions from 0 to 0 and similarly $|\mathcal{A}_{01}|$ is the number of transitions from 0 to 1. The acceptance ratio for the Metropolis-Hastings step is given by

$$\begin{aligned} p_{\text{accept}} &= \min\left(\frac{p(p_{00}^{*} \mid i)q(p_{00}^{(i)} \mid p_{00}^{*})}{p(p_{00}^{(i)} \mid i)q(p_{00}^{*} \mid p_{00}^{(i)})}, 1\right) \\ &= \min\left(\frac{p(i_0 \mid p_{00}^{*})}{p(i_0 \mid p_{00}^{(i)})}, 1\right) \end{aligned}$$

where the simplification here is due to the specific form of the proposal in equation (7.23). Finally, the initial state is assumed to be distributed according to the stationary distribution of the chain, which is given using the standard result from the theory of Markov chains,

$$p(i_0 \mid p_{00}) = \frac{1 - i_0 p_{00} - (1 - i_0)p_{11}}{2 - p_{00} - p_{11}},$$

thus allowing the parameters of the Markov chain prior to be sampled efficiently.

7.5 Marginalized Inference

In [1] impulse and background noise were removed from corrupted audio using the inference method described in section 7.4. However, it is possible to avoid explicit sampling of the z process through marginalization of some of the conditional distributions used in the Gibbs sampler. In this case, the sampling of the Gabor coefficients c, their corresponding indicators γ and the overall noise level σ found in [303] must be modified as shown here. The impulse indicators i and noise scales λ can be sampled from the conditional distributions given in equations (7.12) and (7.16), respectively, as the z process is not required in the conditional distributions given there.

7.5.1 Sampling Gabor Coefficients

The conditional distribution of the Gabor coefficients c is a multivariate Gaussian due to the assumption of Gaussian noise with variance $(1+i_t\lambda_t)\sigma^2$ in the observation y process. For a single c_k coefficient a sample can be drawn jointly with the corresponding indicator variable γ_k from the joint conditional, which can be decomposed as

$$p(c_k, \gamma_k \mid c_{-k}, \gamma_{-k}, y) = p(c_k \mid \gamma, c_{-k}, y)p(\gamma_k \mid \gamma_{-k}, c_{-k}, y), \tag{7.24}$$

where here and throughout what follows dependence of all terms on σ, λ, i and σ_c has been dropped from the notation for brevity.

The Gabor synthesis coefficients $c_k \in \mathbb{R}^2$ are here treated as a vector of two real numbers (corresponding to the coefficient's real and imaginary part), rather than as a single complex one, as described in section 7.1. They can be expected to take a wide range of values, with very large values being comparatively common. The prior chosen for these variables is, therefore, a heavy-tailed Student t distribution, which, as noted above, can be realized as a scale mixture of normals with an inverse gamma mixing distribution, so that

$$p(c_k \mid \sigma_{c_k}, \gamma_k) = (1-\gamma_k)\delta_0(c_k) + \gamma_k \mathcal{N}\left(c_k; 0, \sigma^2_{c_k} I_2\right), \tag{7.25}$$

where I_2 is the 2×2 identity matrix (for the case when $c_k \in \mathbb{R}^2$) and $\sigma^2_{c_k}$ is distributed according to the inverse gamma mixing distribution

$$p(\sigma^2_{c_k} \mid \gamma_k = 1) = \mathcal{IG}(\sigma^2_{c_k}; \kappa, \nu_k). \tag{7.26}$$

Here κ is a shape parameter that determines the heaviness of the tails of the prior distribution. ν_k is a scale parameter that is itself assigned a gamma prior so that

$$\nu_k = f(k)\nu, \tag{7.27}$$

with $\nu \sim \mathcal{G}(\alpha_\nu, \beta_\nu)$ and where $f(k)$ is a fixed weighting function that can be used to express a prior belief about the expected degree of smoothness in the reconstructed signal. The choice of $f(k)$ is discussed in more detail in [303], where the authors suggest using the reciprocal of the frequency modulation number m corresponding to the coefficient k.

In the case when $\gamma_k = 0$, the first term on the right hand side of equation (7.24) does not depend on the observations and so is simply a delta function at the current value of c_k, owing to the prior on c_k in equation (7.25). This means that for coefficients excluded from the reconstruction, c_k need not be updated and that, in this case, the joint distribution to be sampled is given by

$$p(c_k, \gamma_k = 0 \mid c_{-k}, \gamma_{-k}, y) = p(\gamma_k = 0 \mid \gamma_{-k}, c_{-k}, y). \tag{7.28}$$

On the other hand, when $\gamma_k = 1$,

$$\begin{aligned} p(c_k \mid \gamma_k = 1, \gamma_{-k}, c_{-k}, y) &\propto p(y \mid c, \gamma_k = 1, \gamma_{-k}) p(c_k \mid \gamma_k = 1) && (7.29) \\ &\propto \mathcal{N}(y; x, D)\, \mathcal{N}\left(c_k; 0, \sigma^2_{c_k} I_2\right), && (7.30) \end{aligned}$$

with D being a diagonal matrix of noise variances, with the t^{th} diagonal element given by $D_{tt} = (1 + i_t \lambda_t)\sigma^2$, and where x is the true signal, a sample of which is given by the reconstruction in equation (7.3). The reconstruction can be written as the sum of the components that depend on c_k, and those that do not, i.e. $x = \tilde{G}_k \gamma_k c_k + \tilde{G}_{-k} \tilde{c}_{-k}$, allowing the first component on the right of equation (7.30) to be written as

$$\begin{aligned} \mathcal{N}(y; x, D) &= \mathcal{N}\left(y; \tilde{G}_k c_k + \tilde{G}_{-k} \tilde{c}_{-k}, D\right) \\ &= \alpha \mathcal{N}\left(c_k; \frac{\tilde{G}_k' D^{-1}(y - \tilde{G}_{-k} \tilde{c}_{-k})}{\tilde{G}_k' D^{-1} \tilde{G}_k}, (\tilde{G}_k' D^{-1} \tilde{G}_k)^{-1}\right), \end{aligned}$$

using the identity (A.5) in appendix A and where α is a constant of proportionality. Using the identity (A.2) allows equation (7.30) be rewritten in

terms of c_k as

$$p(c_k \mid \gamma_k = 1, \gamma_{-k}, c_{-k}, y) = \mathcal{N}(c_k; \mu_k, \Sigma_k) \tag{7.31}$$

where

$$\begin{aligned} \Sigma_k &= \left(\tilde{G}_k' D^{-1} \tilde{G}_k + \sigma_{c_k}^{-2} I_2\right)^{-1} \\ \mu_k &= \Sigma_k \tilde{G}_k' D^{-1} \left(y - \tilde{G}_{-k} \tilde{c}_{-k}\right). \end{aligned}$$

This allows c_k to be sampled given the corresponding indicator γ_k.

In order to sample from the joint distribution in equation (7.24) the indicator γ_k must be sampled from $p(\gamma_k \mid \gamma_{-k}, c_{-k}, y)$. This cannot be directly evaluated, since the dependency between y and γ_k depends on the value of c_k, and so it is necessary to consider the joint distribution of γ_k and c_k, integrated over all c_k, i.e.

$$\begin{aligned} p(\gamma_k \mid \gamma_{-k}, c_{-k}, z) &= \int p(\gamma_k, c_k \mid \gamma_{-k}, c_{-k}, y) dc_k \\ &\propto p(\gamma_k \mid \gamma_{-k}) \int p(y \mid c, \gamma) p(c_k \mid \gamma_k) dc_k. \end{aligned}$$

The constant of proportionality here is $p(y \mid \gamma_{-k}, c_{-k})$ and, as this does not depend on γ_k is the same for both it possible value of γ_k. Therefore, it suffices to determine the ratio τ_k between these terms when $\gamma_k = 0$ and $\gamma_k = 1$, i.e.

$$\tau_k = \frac{p(\gamma_k = 1 \mid \gamma_{-k}, c_{-k}, y)}{p(\gamma_k = 0 \mid \gamma_{-k}, c_{-k}, y)}, \tag{7.32}$$

and use the fact that the numerator and denominator in equation (7.35) sum to 1 to give

$$p(\gamma_k = 0 \mid \gamma_{-k}, c_{-k}, z) = \tfrac{1}{1+\tau_k}, \tag{7.33}$$

$$p(\gamma_k = 1 \mid \gamma_{-k}, c_{-k}, z) = \tfrac{\tau_k}{1+\tau_k}. \tag{7.34}$$

The ratio τ_k is given, through an application of Bayes' theorem to both numerator and denominator, by

$$\tau_k = \frac{p(\gamma_k = 1 \mid \gamma_{-k}) \int p(y \mid c, \gamma_k = 1, \gamma_{-k}) p(c_k \mid \gamma_k = 1) dc_k}{p(\gamma_k = 0 \mid \gamma_{-k}) \int p(y \mid c, \gamma_k = 0, \gamma_{-k}) p(c_k \mid \gamma_k = 0) dc_k}.$$

The expression inside the integral in the numerator is that on the right hand side of equation (7.30), so the same logic can be followed to derive it, although in this case attention must be paid to the normalizing constants (found in identities (A.2) and (A.5) in appendix A) since the expression is not a probability distributions for c_k and so does not normalize to 1 with respect to c_k as was the case in equation (7.31). In the denominator, $p(y \mid c, \gamma_k = 0, \gamma_{-k}) = \mathcal{N}(y; x_{-k}, D)$, where $x_{-k} = \tilde{G}_{-k}\tilde{c}_{-k}$, the reconstruction of the signal without the k^{th} Gabor atom, and $p(c_k \mid \gamma_k = 0) = \delta_{\{c_k = c_k^{(i)}\}}$, i.e. a delta function at the current sample of c_k. This leads to the expression for τ_k

$$\tau_k = \frac{p(\gamma_k = 1 \mid \gamma_{-k})}{p(\gamma_k = 0 \mid \gamma_{-k})} \frac{|\Sigma_k|^{\frac{1}{2}}}{\sigma_{c_k}^2} \exp\left(\frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k\right), \tag{7.35}$$

which allows the γ_k and c_k to be sampled by first sampling γ_k as a Bernoulli sample with probabilities given by equations (7.33) and (7.34) and then, if this sample for γ_k is 1, sampling c_k from the Gaussian distribution in equation (7.31), but otherwise leaving it unchanged. Note that these distributions apply to the case where the signal is constrained to be real valued and hence $\Sigma_k \in \mathbb{R}^{2\times 2}$, $\mu_k, c_k \in \mathbb{R}^2$, and $\tilde{G}_k \in \mathbb{R}^{L\times 2}$ (given by the corresponding columns of the $\tilde{G}$ matrix).

7.5.2 Sampling Noise Variance

The conditional distribution of the noise variance σ^2 can be found by noting, from the noise model in equation (7.5), that

$$\frac{y_t - x_t}{\sqrt{1 + i_t \lambda_t}} \sim \mathcal{N}\left(0, \sigma^2\right). \tag{7.36}$$

Therefore, given the sample x of the signal reconstruction given by the Gabor coefficients, the scaled and shifted observations on the left of equation (7.36) can be treated as a series of observations of a Gaussian distributed random variable with unknown variance σ^2. The prior on σ^2 can be chosen to be the inverse gamma conjugate prior with distribution $\mathcal{IG}(\alpha, \beta)$. Considering the L scaled and shifted observations, the conditional distribution for σ^2 is

$$p(\sigma^2|y, c, \gamma) = \mathcal{IG}\left(\sigma^2; \alpha + \frac{L}{2}, \beta + \frac{1}{2}\sum_{t=1}^{L} \frac{(y_t - x_t)^2}{1 + i_t\lambda_t}\right).$$

Unless something is known about the scale of the noise in advance, the parameters α and β should be chosen to give a vague prior on σ^2 (see figure 7.4).

7.6 Results

In order to evaluate the methods their ability to restore audio tracks distorted with both artificial noise generated from the noise model and with real noise taken from the run-in track of an old vinyl recording was compared. To generate the artificially noisy track, a clean track was corrupted by adding noise from the model in equation 7.5. The presence of impulses was modelled as following a Markov chain process as described in section 7.3, with $p_{00} = 0.995$ and $p_{11} = 0.9$. Impulse scale λ_t was modelled as either being fixed to $\lambda_t = 100$ for all t ('Fixed λ'), or being drawn from the prior in equation (7.6), with $\alpha_\lambda = 1$ and $\beta_\lambda = 20$ ('Variable λ'), giving roughly the same SNR as in the fixed case . For the 'real' noise, a multiple of the run-in track of an old vinyl recording was added to the clean track. This run-in track was high-pass filtered to remove low-frequency distortion not removed by these algorithms, and which otherwise has a dominant effect on the SNR (since it effectively moves the signal's zero level), making comparison difficult. The signal to noise ratio of a reconstruction (or distorted

signal) $\hat{x}$ is calculated in dB as

$$\mathrm{SNR}_{\mathrm{dB}} = 10 \log_{10} \left(\frac{\sum_t x_t^2}{\sum_t (x_t - \hat{x}_t)^2} \right). \tag{7.37}$$

Table 7.1, taken from [1], shows the effect of these various impulse models. For the results in this table, 200 MCMC samples were generated, with the first 100 samples being considered burn in; from the convergence results shown below this appears adequate. All results in this table were produced using an explicit z process initialized to the observed signal y. In the case of variable scale impulses in the data used with a fixed variance impulse model for their removal, the impulse variance parameter λ_{fixed} was set to the mean impulse variance. From these results it is clear that if impulses are present then having a model that takes account of them makes a substantial impact on restoration performance. The restoration of audio corrupted with real vinyl noise was also substantially improved, suggesting that impulses are present in this noise, or at least that the impulse noise model is a better fit for this data than the homogenous noise model (the 'None' impulse model in table 7.1). As might be expected, the variable impulse method performs slightly worse for fixed impulses with known variance, as the fixed impulse algorithm in that case is correctly tuned to the impulse size. For variable scale artificial impulses and real impulses the variable impulse model gives the greatest improvement in SNR with no need for tuning of the impulse size. On the other hand, though the fixed impulse model produced good results when impulse scale was well tuned, the value of the scale parameter λ chosen had a sizeable impact on performance.

In order to compare the algorithm with marginalized z process to that with an explicitly sampled z process, a similar test (with different data) was run with each algorithm and the original ('no impulses') algorithm applied to tracks with both artificial (variable λ_t) and real noise. The results were generated using a simple Bernoulli prior for the presence of Gabor

Impulse Model	Impulses in Data	Noisy SNR (dB)	Final SNR (dB)
None	Fixed λ	6.97	12.32
None	Variable λ	6.96	9.65
None	Real	6.61	8.83
Fixed λ	Fixed λ	6.97	13.83
Fixed λ	Variable λ	6.96	13.36
Fixed $\lambda = 15$	Real	6.61	11.54
Fixed $\lambda = 100$	Real	6.61	12.79
Variable λ	Fixed λ	6.97	13.36
Variable λ	Variable λ	6.96	13.47
Variable λ	Real	6.61	12.81

Table 7.1: SNRs (dB) before and after noise removal assuming a range of models of the impulses present in the signal, and with impulses in the data deriving from a range of different sources

	Artificial Noise (dB)	Real Noise (dB)
Distorted signal	6.10	7.10
No impulses	10.34	8.98
Marginalized	17.55	12.40
Sampled z process	17.48	12.25

Table 7.2: SNRs (dB) before and after noise removal with sampled z process, marginalized z process and assuming no impulses (as a baseline), for artificial and real impulses. The top line ('Distorted signal') shows the SNR of the signal before restoration

components, with $p = 0.5$. Table 7.2 shows the results of this in terms of reconstruction SNR. Figures 7.7-7.9 show the reconstruction of distorted audio signals using the marginalized and z sampling algorithms. The resulting reconstructions are very similar; the reconstruction using sampled z is almost completely obscured by that using marginalized z in figures 7.7 and 7.8, and the SNRs achieved in table 7.2 are almost identical for the two algorithms. For the artificial data (where impulse positions are known), impulse detection is good, with both algorithms picking 89-90% of impulses; those missed are of often of low intensity. Figure 7.9 shows a detail from figure 7.7, allowing some properties of the reconstruction to be better observed. In this period all true impulses are detected, but there are also some false detections in positions where the true audio track contained sharp changes. This illustrates an inherent problem with noise reduction: in places where the true signal appears similar to the noise being removed (such as the large peak between samples 7.115 and 7.2×10^4 in figure 7.9), noise reduction introduces distortion by removing those features. Of course, if the noise level is substantial and the noise can be usefully characterized as happens here, noise reduction still offers a significant benefit.

An interesting difference between the results with artificial and real noise is that in the latter case much more background 'hiss' is left in the reconstruction. This can be seen in a comparison of figures 7.7 and 7.8 (which aim to reconstruct the same underlying signal), where, for example around sample 1×10^4, the noise in the latter reconstruction is higher than that in the former. It is also visible in the top left plot in figure 7.11 where the noise level σ^2 converges to a significantly higher value for the artificial noise than for the real noise. This is likely to be because the noise model is not a perfect fit for the real noise. To see this, figure 7.10 compares the distribution of the real and artificial noise used in these two examples. In the distribution of the artificial noise there is a distinct 'kink' at an absolute size of about 0.1 not present in the distribution of the real noise, the frequency of which decreases more smoothly as impulse size increases. This reflects

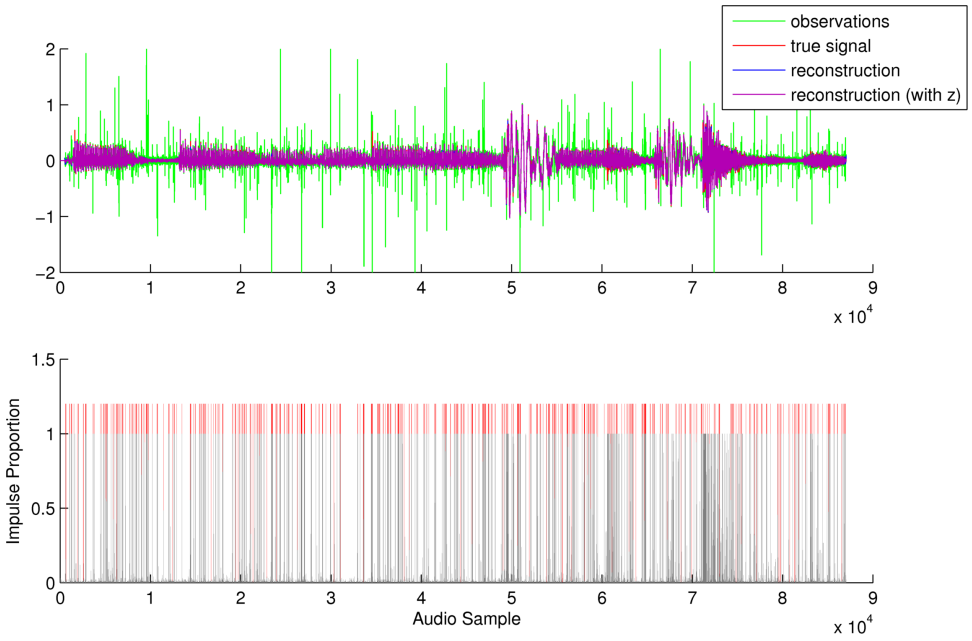


Figure 7.7: Top: Reconstruction of an audio track distorted with artificial noise generated from the noise model using algorithms with marginalized and sampled z processes. Bottom: Impulse detection showing true impulse positions (red) along with proportion of samples containing an impulse at each time period (grey)

the distinction between impulse and background noise in the model; the distribution of the artificial noise is effectively a mixture of a Gaussian and t-distribution as discussed in section 7.4. The real noise is also somewhat more heavy-tailed. Analysis of the distribution of real impulses might offer a way of choosing the α_λ and β_λ prior parameters governing the impulse process, attempting to better match the model noise distribution with that of the real impulses. This has not been investigated further here. If there is no strong distinction between impulses and background noise in real data, this could lead to a model in which the variable scale 'impulse' noise model was used for most samples, leading to a system for effectively removing t-distributed background noise.

Figures 7.11-7.14 illustrate the convergence rates of the marginalized and sampled z algorithms. These appear to show that for both real and

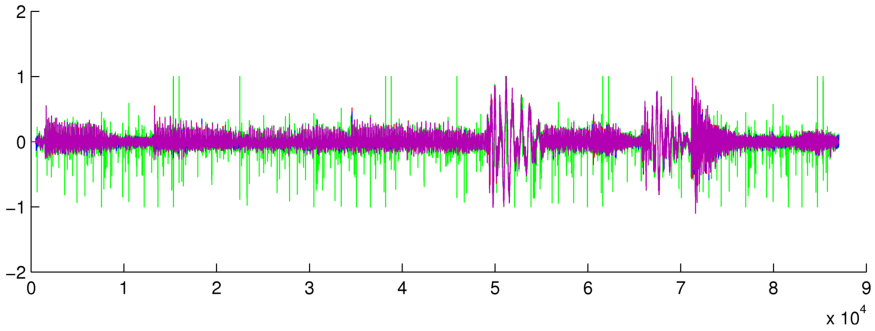


Figure 7.8: Reconstruction of audio signal corrupted by real impulse noise using both marginalized and sampled z processes. Line colours as in figure 7.7

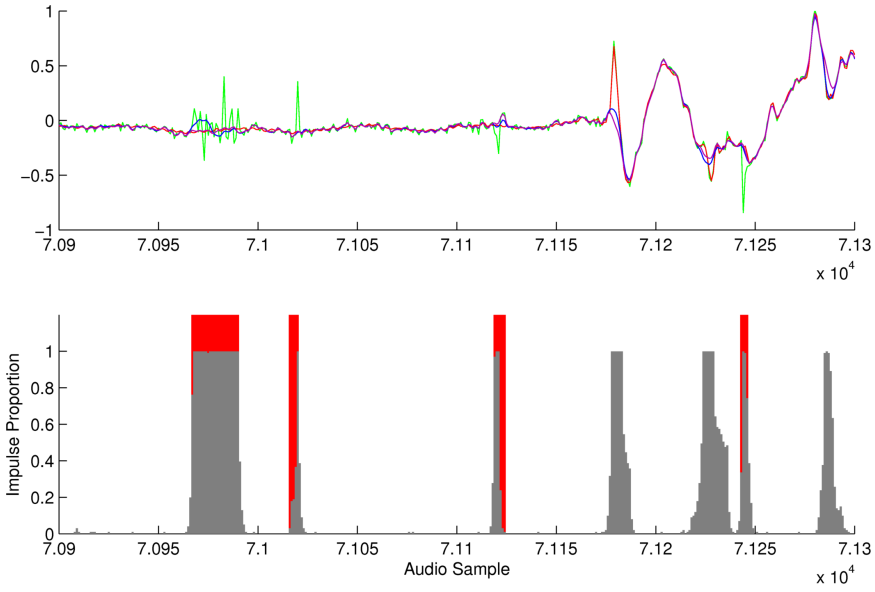


Figure 7.9: Detail of figure 7.7 over 400 audio samples (about 0.01s of audio)

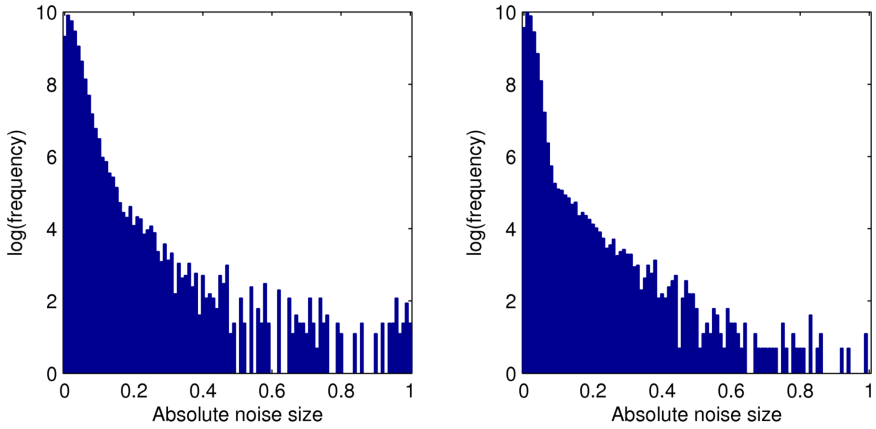


Figure 7.10: Histogram of absolute noise size ($|y_t - x_t|$) for track distorted with real noise (left) and artificial noise (right)

artificial noise, the algorithm converges rapidly, with a 100 sample burn-in period being acceptable for the results here. Perhaps surprisingly, the marginalized algorithm does not seem to offer any noticeable improvement in convergence speed over explicitly sampling the z process. When real noise is present, convergence was slightly slower than that with artificial noise; this is probably due to the noise model not being a perfect fit for the observed noise.

Figures 7.13 and 7.14 show the estimates of the prior parameters p_{00} and p_{11} governing the prior on impulse presence. Figure 7.13 shows rapid convergence close to the correct values for artificial data. The slight underestimation of p_{00} seen here could be due to false detection impulses due to the shape of the original signal as discussed above. For real impulses, values of $p_{00} \approx 0.985$ and $p_{11} \approx 0.835$ were obtained, both slightly lower than those used for the artificial noise, suggesting the presence of more frequent but shorter impulses in the real noise.

It is difficult to make a full assessment of the results based only on the SNR and a more complete evaluation would include a psychoacoustical metric. Some distortion (ringing) due to the filtering can be evident in badly corrupted examples with artificial noise; from a perceptual point of view this can be improved by actually slightly increasing the amount of

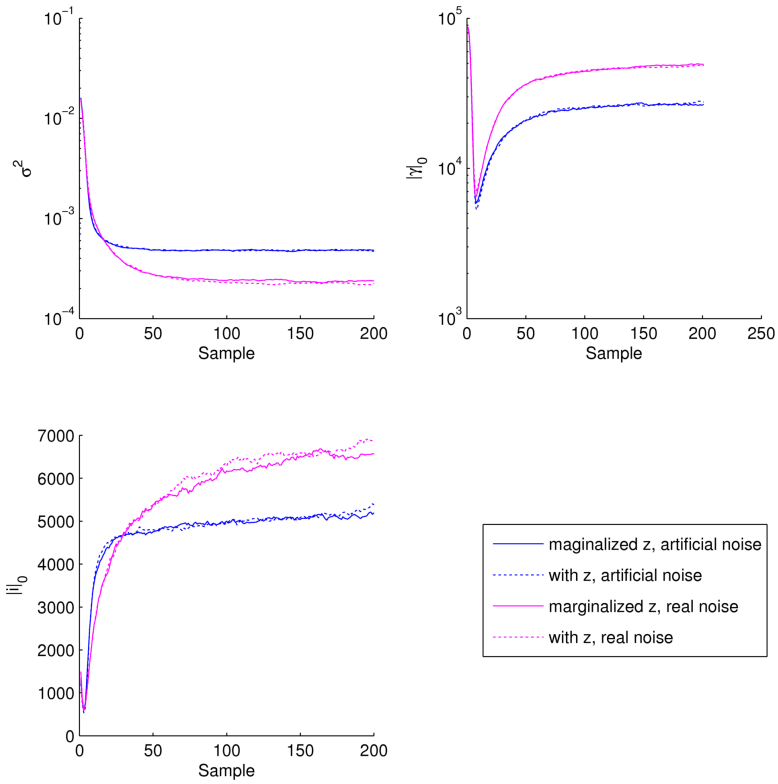


Figure 7.11: Comparison of convergence of noise level σ^2 (top left), number of non-zero Gabor coefficients $|\gamma|_0$ (top right), and number of detected impulses $|i|_0$ (bottom left) for marginalized (solid) and sampled z (dotted), with artificial (blue) and real (purple) noise

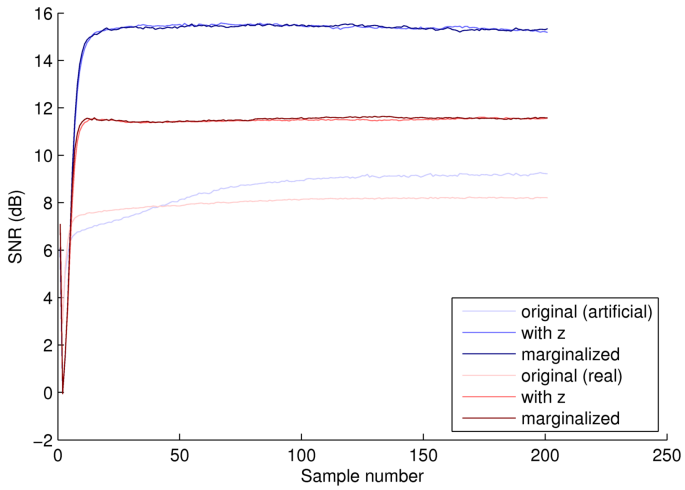


Figure 7.12: Signal to noise ratio with sample number for original (no impulses), sampled z and marginalized z algorithms with artificial (blue) and real (red) noise

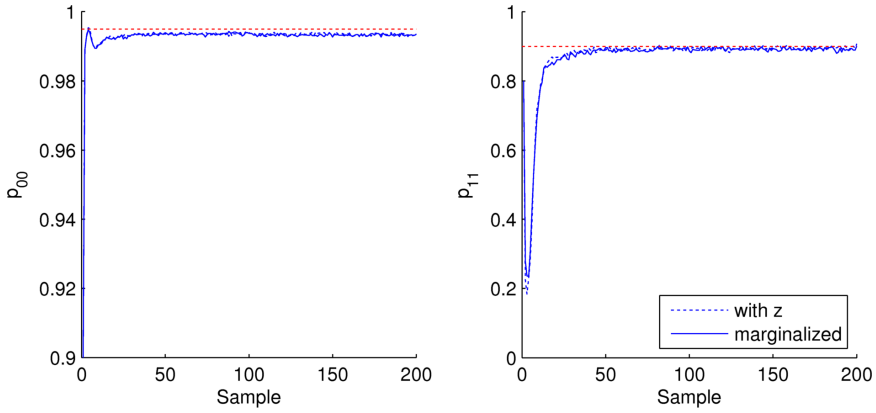


Figure 7.13: Estimated parameters p_{00} and p_{11} of impulse presence Markov chain for artificial noise; true values shown as dotted red line

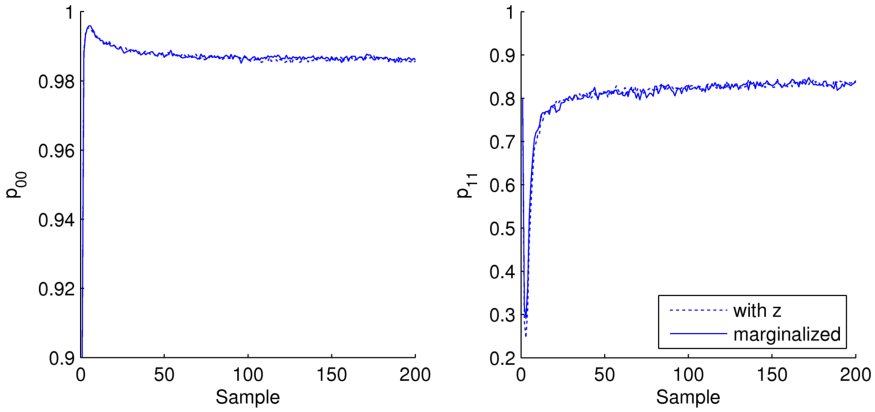


Figure 7.14: Estimated parameters p_{00} and p_{11} of impulse presence Markov chain for real noise

noise in the final reconstruction. On the other hand, with real noise, the algorithm does not always remove enough background noise (although perceptually, impulses are successfully removed) and this might be due to an incorrect or badly configured noise model.

7.7 Conclusions

This chapter has outlined a way to simultaneously remove impulse and background noise from corrupted audio signals. The methods presented extend the work of [303] by adding the capacity to remove impulse noise, and [1] by showing how the artificial z process introduced there does not need to be sampled. This latter extension does not appear to have a significant impact on performance and, in particular, does not appear to improve convergence of the MCMC sampler.

Impulses with a wide range of sizes can be removed and the algorithm was shown to significantly improve the signal-to-noise ratio for signals distorted with both artificially generated noise, and with real noise taken from an old vinyl recording. For these latter impulses the algorithm tended to leave some background noise in the signal. This could be perhaps be improved by further work to tune the model to the characteristics of real

noise.

The algorithm could be extended to a sequential setting, using particle filtering, which, if sufficiently efficient, would allow real-time noise reduction. Applied only to estimation of the Gabor and impulse coefficients (and indicators), such an algorithm could be combined with the Gibbs sampling of parameters shown here to offer an alternative method of batch noise reduction and parameter estimation, via a Particle Gibbs scheme, although it is not clear that this would offer a substantial advantage over the MCMC scheme used here.

Chapter 8

Conclusion

The aim of this work was to develop methods for Bayesian inference for a range of systems, motivated by examples in finance, physical object tracking and audio restoration. In each of these areas, a method has been presented that extends the state of the art in some way. Of the methods developed, the most important contributions are perhaps in the area of model estimation, with new models allowing efficient, Bayesian solutions introduced for the learning of environment structure in a SLAM-like problem and for the learning of inter-object group structure from object tracking data. The method of inferring system structure from successive state estimates using an inverted numerical integrator and approximating intractable terms with a noise term is widely applicable.

In chapter 3, an efficient method for state estimation was given for jump-diffusion systems with linear Gaussian diffusions. In chapter 4 it was shown how this technique could be used directly for Bayesian parameter inference amounting to system identification in such systems. An alternative Bayesian parameter estimation method based on reversible-jump MCMC was also introduced.

For tracking problems in unknown environments a method was introduced in chapter 5 that allowed the unknown environment structure to be mapped in a way similar to mapping in SLAM problems. It made the

mapping of quite general environment structures due to the use of non-parametric priors on a potential field model of environment structure. The method works sequentially and allows efficient solution using particle filtering, although in the form presented in chapter 5 suffers from computational costs that increase rapidly with the number of observations, making it unsuitable for long time series. This latter problem can be solved through the use of sparse Gaussian process approximations and a map divided into a grid, as described there.

Though not yet sequential, the method introduced for group structure inference in chapter 6 offers a new way for learning about the structure of networks of interacting entities. The method is applicable both in the physical object tracking context in which it is introduced here, and more widely such as for the inference of gene regulatory networks and for learning about relationships between indicators in econometrics. In these applications, the ability of the method to make causal inference is likely to be of interest, as is the ability to infer sparse network structures.

Chapter 7 illustrated how sparse model structures could be used in audio signal processing, in particular in the removal of impulse noise such as the pops and clicks found on old vinyl recordings. The method is effective for impulse removal, although it is computationally demanding and only able to run in batch form. A more efficient marginalized version of the algorithm that originally appeared in [1] was presented.

8.1 Recommendations for Future Work

There are many ways in which the work in this thesis could be extended and each chapter makes some recommendations for further investigation that could be undertaken. This section briefly lists some of the most promising areas of further investigation, many of which will it is hoped will be able to be undertaken in the near future.

Jump-diffusion models are common models of electricity spot prices,

and so it would be interesting to compare the parameter estimates obtained using the methods in chapters 3 and 4 to those derived elsewhere, especially since most existing estimation techniques used on this data only give point parameter estimates. Generalizing the models in chapters 3 and 4 to linear diffusion systems with non-linear observation functions may also be of interest, though this will eliminate some of the efficiency benefits of Rao-Blackwellization.

The combination of the variable rate particle filter and Particle MCMC methods produces an alternative to reversible jump MCMC for certain types of variable dimension time series systems that might be of interest for batch estimation in a number of areas, including physical object tracking, econometrics and finance. Whether this method can be generalized to other more general variable-dimension estimation problems is not yet clear and merits further investigation.

The algorithm introduced for the mapping of unknown structured environments in chapter 5 can be extended to work for much longer time series over larger areas through the use of sparse Gaussian process methods and a grid structure for the map, as suggested in that chapter. This is essential to make the system applicable in real tracking applications and is an urgent priority in the further development of this method. For long data series, it is likely that the particle history degeneracy problem will have an impact on the performance of the mapping algorithm, and ways of mitigating this effect should be investigated further.

There are many ways in which the work in chapter 6 on the learning of group structure could be extended and applied. The development of a sequential scheme now seems possible and this would allow the method to be directly applied in on-line tracking applications. The mechanism for the inference of sparse structure mentioned in chapter 6 still needs more refinement and testing; fast, accurate methods for this would allow efficiency improvements in the case of disjoint groups and useful approximations in the case of nearly disjoint groups.

A number of applications for the group inference system are also of interest. By adding velocity based relationships, the model presented in chapter 6 can be extended to encompass standard flocking rules. It would be interesting to see whether such rules could be recovered from real animal behaviour. Related models have already been applied to gene regulatory networks and comparison with those results should indicate whether the model here could produce useful results in that domain. It would also be particularly interesting to apply the system to collections of economic indicators to see if non-linear causal relationships could be inferred.

A number of the methods presented in this thesis make use of the numerical integration of non-linear Langevin equations (or could be adapted to do so), using the integration method in appendix D. This is only one of a number of methods available for such equations, and a more thorough investigation of the various methods available and their suitability for various tasks would be a useful future piece of work. Since most of these methods come from the physics literature, a review of these methods with respect to their use in Bayesian tracking and smoothing problems might also be of wider interest.

Appendix A

Useful Gaussian Identities

This appendix contains a number of useful identities involving manipulations of the Gaussian distribution and multivariate Gaussian distribution that are used throughout this thesis.

A.1 Affine Argument Transform

$$\mathcal{N}(Ax + b; \mu, \Sigma) = \frac{1}{|A|}\mathcal{N}\left(x; A^{-1}(\mu - b), A^{-1}\Sigma(A^{-1})'\right) \tag{A.1}$$

where x, b, μ are $k \times 1$; A, Σ are $k \times k$; A is invertible, Σ symmetric positive definite.

A.2 Product of Two Multivariate Gaussian PDFs

$$\mathcal{N}(x; \mu_1, \Sigma_1)\,\mathcal{N}(x; \mu_2, \Sigma_2) = \alpha\mathcal{N}(x; \mu, \Sigma) \tag{A.2}$$

with

$$\begin{aligned}\alpha &= \frac{|\Sigma|^{\frac{1}{2}}}{(2\pi)^{\frac{k}{2}}|\Sigma_1|^{\frac{1}{2}}|\Sigma_2|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}\left(\mu_1'\Sigma_1^{-1}\mu_1+\mu_2\Sigma_2^{-1}\mu_2-\mu'\Sigma^{-1}\mu\right)\right]\\ \Sigma^{-1} &= \Sigma_1^{-1}+\Sigma_2^{-1}\\ \mu &= \Sigma\left(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2\right)\end{aligned}$$

where x, μ, μ_1, μ_2 are $k \times 1$; $\Sigma, \Sigma_1, \Sigma_2$ are $k \times k$ symmetric positive definite.

A.3 Quotient of Two Multivariate Gaussian PDFs

$$\frac{\mathcal{N}(x;\mu_1,\Sigma_1)}{\mathcal{N}(x;\mu_2,\Sigma_2)} = \alpha\mathcal{N}(x;\mu,\Sigma) \tag{A.3}$$

with

$$\begin{aligned}\alpha &= \frac{(2\pi)^{\frac{k}{2}}|\Sigma|^{\frac{1}{2}}|\Sigma_2|^{\frac{1}{2}}}{|\Sigma_1|^{\frac{1}{2}}}\exp\left[-\frac{1}{2}\left(\mu_1'\Sigma_1^{-1}\mu_1-\mu_2\Sigma_2^{-1}\mu_2-\mu'\Sigma^{-1}\mu\right)\right]\\ \Sigma^{-1} &= \Sigma_1^{-1}-\Sigma_2^{-1}\\ \mu &= \Sigma\left(\Sigma_1^{-1}\mu_1-\Sigma_2^{-1}\mu_2\right)\end{aligned}$$

where x, μ, μ_1, μ_2 are $k \times 1$; $\Sigma, \Sigma_1, \Sigma_2$ are $k \times k$ symmetric positive definite.

A.4 Product of Univariate Gaussian PDFs

$$\prod_{i=1}^{k}\mathcal{N}\left(x;\mu_i,\sigma_i^2\right) = \alpha\mathcal{N}\left(x;\mu,\sigma^2\right) \tag{A.4}$$

with

$$\begin{aligned}
\alpha &= \frac{\sigma}{(2\pi)^{\frac{k-1}{2}} \prod_i \sigma_i} \exp\left[-\frac{1}{2}\left(\sum_i \frac{\mu_i^2}{\sigma_i^2} - \sigma^2\left(\sum_i \frac{\mu_i}{\sigma_i^2}\right)^2\right)\right] \\
\sigma^2 &= \left(\sum_i \frac{1}{\sigma_i^2}\right)^{-1} \\
\mu &= \sigma^2 \sum_i \frac{\mu_i}{\sigma_i^2}
\end{aligned}$$

where all quantities are scalar.

A.5 Linearly Dependent Elements

$$\mathcal{N}(ax; c, B) = \alpha \mathcal{N}\left(x; \mu, \sigma^2\right) \tag{A.5}$$

with

$$\begin{aligned}
\alpha &= \frac{\sigma}{(2\pi)^{\frac{k-1}{2}} |B|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\left(c'B^{-1}c - \frac{(a'B^{-1}c)^2}{a'B^{-1}a}\right)\right] \\
\sigma^2 &= (a'B^{-1}a)^{-1} \\
\mu &= \frac{a'B^{-1}c}{a'B^{-1}a} = \sigma^2 a'B^{-1}c
\end{aligned}$$

where x is 1×1; a, c are $k \times 1$; B is $k \times k$ symmetric positive definite.

Appendix B

Matrix Fraction Decomposition

This appendix shows how to calculate the covariance of the LTI system

$$dX = AXdt + BdW_t$$

appearing in chapter 3, using Matrix Fraction Decomposition [177].

The required covariance is given by equation (3.7) as

$$\text{cov}(X_T) = e^{A(T-S)}\Big[Q(S,T) + \text{cov}(X_S)\Big](e^{A(T-S)})',$$

with

$$Q(r,s) = \int_r^s e^{-At}BB'(e^{-At})'dt$$

Here, it will be shown how to calculate the $e^{A(T-S)}Q(S,T)(e^{A(T-S)})'$ component, taking, without loss of generality $S = 0$. Let

$$P(T) = e^{AT}Q(0,T)(e^{AT})' = \int_0^T e^{A(T-t)}BB'(e^{A(T-t)})'dt.$$

Differentiating $P(T)$ with respect to T gives

$$\frac{dP}{dT} = AP + PA' + BB'. \tag{B.1}$$

This matrix differential equation can be solved by taking $P = CD^{-1}$ where

$$\begin{bmatrix} {}^{dC}/_{dT} \\ {}^{dD}/_{dT} \end{bmatrix} = \begin{bmatrix} A & BB' \\ 0 & -A' \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix}, \tag{B.2}$$

and thus

$$\begin{aligned} {}^{dC}/_{dT} &= AC + BB'D \\ {}^{dD}/_{dT} &= -A'D. \end{aligned}$$

This can be seen by substituting these relations into equation (B.1) so that

$$\begin{aligned} \frac{dP}{dT} &= \frac{d(CD^{-1})}{dT} \\ &= C\frac{dD^{-1}}{dT} + \frac{dC}{dT}D^{-1} \\ &= -CD^{-1}\frac{dD}{dT}D^{-1} + \frac{dC}{dT}D^{-1} \\ &= -CD^{-1}(-A'D)D^{-1} + D^{-1}AC + BB'DD^{-1} \\ &= PA' + AP + BB', \end{aligned}$$

as required. In the third line the following identity has been used,

$$\frac{dD^{-1}}{dT} = -D^{-1}\frac{dD}{dT}D^{-1},$$

which can be seen from the matrix chain rule $d(XY) = XdY + dXY$.

In the case considered here, $P(0) = 0$. The matrix differential equation in equation (B.2) has the form ${}^{dX}/_{dT} = AX$, so has the general solution $X = e^{AT}X_0$. Since $P(0) = C(0)D(0)^{-1}$, initial conditions $C(0) = P(0)$ and $D(0) = I$ can be chosen, allowing equation (B.2) to be solved as

$$\begin{bmatrix} C(T) \\ D(T) \end{bmatrix} = \exp\left(\begin{bmatrix} A & BB' \\ 0 & -A' \end{bmatrix} T\right) \begin{bmatrix} C(0) \\ D(0) \end{bmatrix},$$

which is a standard calculation and which yields $P(T) = C(T)(D(T))^{-1}$.

Appendix C

PMCMC Derivations

Let

$$D = p(x_0^{b_0})\nu_T^k \prod_{t=0}^{T} N_t \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})$$

then,

$$\begin{aligned} D &= p(x_0^{b_0})\nu_T^k \prod_{t=0}^{T} N_t \prod_{t=0}^{T-1} R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t}) \frac{p(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})}{p(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})} \frac{\nu_t^{b_t}}{\nu_t^{b_t}}, \\ &= p(x_{0:T}^k)\nu_T^k \prod_{t=0}^{T} N_t \prod_{t=0}^{T-1} \nu_t^{b_t} \left(\frac{R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})}{p(x_{t+1}^{b_{t+1}} \mid x_t^{b_t}) \nu_t^{b_t}} \right). \end{aligned}$$

Noting that $\nu_t^{b_t}$ is the normalized weight (except when $t = 0$, when it is $1/N_0$), this is

$$\begin{aligned} D &= p(x_{0:T}^k) \prod_{t=1}^{T} \frac{N_t w_t^{b_t}}{\sum_i w_t^i} \prod_{t=0}^{T-1} \left(\frac{R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})}{p(x_{t+1}^{b_{t+1}} \mid x_t^{b_t}) \nu_t^{b_t}} \right), \\ &= p(x_{0:T}^k) \frac{1}{\prod_{t=1}^{T} \frac{1}{N_t} \sum_i w_t^i} \prod_{t=0}^{T-1} \left(w_{t+1}^{b_{t+1}} \frac{R(b_t \mid \nu_t) q(x_{t+1}^{b_{t+1}} \mid x_t^{b_t})}{p(x_{t+1}^{b_{t+1}} \mid x_t^{b_t}) \nu_t^{b_t}} \right). \end{aligned}$$

The denominator of the first fraction is the approximate (estimated) likelihood from equation (4.8), and using the definition of the weights in equa-

tion (4.7), this expression simplifies to

$$\begin{aligned} D &= \frac{p(x_{0:T}^k)}{\hat{p}(y_{1:T})} \prod_{t=0}^{T-1} p(y_{t+1} \mid x_{1:T}^k) \\ &= \frac{p(x_{0:T}^k) p(y_{1:T} \mid x_{0:T}^k)}{\hat{p}(y_{1:T})}, \end{aligned}$$

as given in equation (4.13).

Appendix D

Numerical Solution of Langevin SDEs

Chapters 5 and 6 deal with systems whose dynamics can be modelled by non-linear Langevin stochastic differential equations. These models do not have analytical solutions for their density evolution and so numerical integration is necessary in order to derive the corresponding state transition densities required for inference. This appendix outlines the numerical integration scheme used in those chapters. The SDEs of interest are Langevin equations of the form

$$d^2\mathbf{x} = f(\mathbf{x})dt + g(\mathbf{x})dW_t, \tag{D.1}$$

where dW_t is the infinitesimal change of a standard Brownian motion at time t. This can be rewritten as a system of two first order SDEs, introducing a velocity variable $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}}dt = d\mathbf{x} \tag{D.2}$$

$$d\dot{\mathbf{x}} = f(\mathbf{x})dt + g(\mathbf{x})dW_t. \tag{D.3}$$

These SDEs are really just shorthand for integral equations, i.e.

$$\begin{aligned}\int \dot{\mathbf{x}} dt &= \int d\mathbf{x} \\ \int d\dot{\mathbf{x}} &= \int f(\mathbf{x}) dt + \int g(\mathbf{x}) dW_t,\end{aligned}$$

and it is these integrals that are approximated numerically.

Langevin equations occur in statistical physics and chemistry and their numerical integration has been studied in that context, especially with respect to the simulation of Langevin systems in molecular dynamics. Early approaches to the simulation of these equations used forward Euler-Maruyama schemes [318] (essentially just Euler schemes applied to stochastic equations, see e.g. [319]), but since then a number of improved schemes have been proposed. These include that of [320], a third order velocity-free scheme related to Verlet integrators for non-stochastic systems, and a series of methods based on Runge-Kutta integrators adapted for the stochastic case, for example the second order scheme in [321]. Such schemes suffer from the need for multiple evaluations of the force term $f(\mathbf{x}_t)$, which tends to be the main computational cost of these methods, especially in the constant noise case $g(\mathbf{x}_t) = \sigma$.

Higher order schemes based on the 'stochastic expansion' [322] of the SDEs are also available. These have the disadvantage of requiring the evaluation of derivatives of the force terms unlike stochastic Runge-Kutta schemes. However, the Gaussian process approximations used in chapters 5 and 6 make this quite straightforward, although the question of its numerical stability remains. A number of fourth-order schemes of this type are available, including [323], upon which the scheme in this appendix is based; see e.g. [322], [324] and [325] and the references therein.

In [322], Euler-Maruyama, stochastic Runge-Kutta and stochastic expansion methods are compared. There it is concluded that stochastic expansion methods offer an advantage over Runge-Kutta and Euler methods in terms of accuracy, but that Runge-Kutta methods may offer better com-

putational efficiency due to their simpler form and avoidance of derivative evaluation for systems of many objects. Another comparison is found in [325] with a particular emphasis on the accuracy of the approximation of the stationary density, with methods compared using their approximation of the exactly solvable linear stochastic oscillator (see section D.6). It is found that both leapfrog and the fully implicit midpoint Euler method (a Runge-Kutta scheme) correctly find the state distribution in the linear case, although these latter methods are expensive in the non-linear case due to their implicit form, which requires the solution of a non-linear system. Leapfrog methods require velocity and position estimates at offset times, so require a slight modification of the inference scheme in order to accommodate them.

The scheme presented in this appendix is based on that in [323], although here only the lower-order parts of that scheme are used. This leaves it somewhat similar to the scheme in [320], at least in the position component. The scheme in [323] was chosen partly because of its intuitive derivation and the ease with which it could be truncated to give a scheme with a simple functional dependence on the value of $f(\mathbf{x}_t)$, especially in the position component. This is used for (partial) system identification in chapters 5 and 6. However, it is clear that such approximations can be found for other schemes; indeed some schemes not involving derivatives might be better suited to this. The basic idea of these approximations is to rearrange the scheme to give an expression for $f(\mathbf{x}_t)$ in terms of the system state, with system noise and intractable terms in the expansion approximated as a single noise term. This gives a noisy 'pseudo-observation' of the function at a particular position $f(\mathbf{x}_t)$, which can be used to build an estimate of its form.

In chapters 5 and 6, the same truncated numerical scheme has been used for both system identification and in forward simulation. However, use of the same integrator in both applications is *not* necessary and thus the more accurate schemes available might, in fact, be better choices for forward simulation. The use of more accurate integration schemes might

improve the performance of the methods in those chapters and thus these schemes warrant further investigation, although this has not been pursued further here.

The focus in this appendix is on the development of a single-step scheme, but an alternative is to use multi-step schemes, in which the state values at a series of (unobserved) intermediate points are calculated. These methods are briefly discussed in section D.5. A book-length review of the numerical integration of general SDEs is given in [176], and [319] gives a good introduction to the area.

This appendix is structured as follows. Section D.1 gives some simple Euler-Maruyama schemes for Langevin SDEs. Section D.2 derives the univariate and multivariate higher order schemes used in chapters 5 and 6. Section D.3 gives transition densities for the schemes in section D.2 in the case of random and non-random force terms. Section D.4 discusses the trade-off that exists in particle filtering algorithms between more accurate numerical integration schemes for which these densities are intractable and less accurate schemes with tractable densities. Section D.5 briefly introduces multi-step schemes, in particular the Bayesian imputations approach. Finally, section D.6 compares three versions of the higher order scheme developed with the Euler schemes in section D.1.

D.1 Euler-Maruyama Schemes

The simplest approach to numerical integration is the forward Euler-Maruyama scheme, which gives the approximation

$$\dot{\mathbf{x}}_{t+h} = \dot{\mathbf{x}}_t + f(\mathbf{x}_t)h + g(\mathbf{x}_t)Z_t \tag{D.4}$$

$$\mathbf{x}_{t+h} = \mathbf{x}_t + \dot{\mathbf{x}}_t h, \tag{D.5}$$

where $Z_t \sim \mathcal{N}(0, hI)$. This is simply the standard Euler scheme applied to a stochastic system. It is a fully explicit method, with all necessary quantities for its calculation available at time t. Unfortunately using this scheme for

single-step integration from a known starting value leads to a point estimate of the particle's position at $t + h$, which makes it unsuitable for use in approximating a transition density, or at least likely to lead to difficulties with many Bayesian inference methods.

A simple modification that gives a non-degenerate state distribution at $t + h$ is to use a two-step scheme, where a step of the forward Euler scheme to $t + h/2$ is calculated first, followed by a second step from $t + h/2$ to $t + h$, giving

$$\begin{aligned}
\dot{\mathbf{x}}_{t+h/2} &= \dot{\mathbf{x}}_t + h/2 f(\mathbf{x}_t) + g(\mathbf{x}_t) Z_t \\
\mathbf{x}_{t+h/2} &= \mathbf{x}_t + h/2 \dot{\mathbf{x}}_t \\
\dot{\mathbf{x}}_{t+h} &= \dot{\mathbf{x}}_{t+h/2} + h/2 f(\mathbf{x}_{t+h/2}) + g(\mathbf{x}_{t+h/2}) Z_{t+h/2} \\
\mathbf{x}_{t+h} &= \mathbf{x}_{t+h/2} + h/2 \dot{\mathbf{x}}_{t+h/2},
\end{aligned}$$

with Z_t and $Z_{t+h/2}$ both being distributed as $\mathcal{N}(0, h/2 I)$. This integration scheme gives the predictive distribution of $\mathbf{x}_{t+h}$ as

$$p(\mathbf{x}_{t+h} | \mathbf{x}_t, \dot{\mathbf{x}}_t, U) \sim \mathcal{N}\left(\mathbf{x}_t + \dot{\mathbf{x}}_t h + \frac{h^2}{4} f(\mathbf{x}_t), \frac{h^3 g^2(\mathbf{x}_t)}{8}\right),$$

but the predictive distribution of $\dot{\mathbf{x}}_{t+h}$ is non-Gaussian (except in the linear case), due to its dependence on $f(\mathbf{x}_{t+h/2})$, with $\mathbf{x}_{t+h/2}$ itself being a (Gaussian) random variable. In the tests in section D.6 this scheme tends to underestimate the variance of the particle's position $\mathbf{x}_{t+h}$.

A second simple modification is to use a *semi-implicit Euler* scheme, where the $t + h$ velocity is used in the standard Euler update:

$$\begin{aligned}
\dot{\mathbf{x}}_{t+h} &= \dot{\mathbf{x}}_t + f(\mathbf{x}_t) h + g(\mathbf{x}_t) Z_t \\
\mathbf{x}_{t+h} &= \mathbf{x}_t + \dot{\mathbf{x}}_{t+h} h
\end{aligned} \tag{D.6}$$

where $Z_t \sim \mathcal{N}(0, hI)$. The scheme is called semi-implicit because it makes use of $\dot{\mathbf{x}}_{t+h}$ to estimate $\mathbf{x}_{t+h}$, but uses $\mathbf{x}_t$ rather than $\mathbf{x}_{t+h}$ to estimate $\dot{\mathbf{x}}_{t+h}$, which would make the scheme fully implicit. The semi-implicit scheme,

unlike the fully implicit scheme is straightforward to calculate for Langevin equations due to the simple form of equation (D.2). It gives the predictive distribution of $\mathbf{x}_{t+h}$ as

$$p(\mathbf{x}_{t+h}|\mathbf{x}_t, \dot{\mathbf{x}}_t, U) \sim \mathcal{N}\left(\mathbf{x}_t + \dot{\mathbf{x}}_t h + h^2 f(\mathbf{x}_t), h^3 g^2(\mathbf{x}_t)\right),$$

and that of $\dot{\mathbf{x}}_{t+h}$ as

$$p(\dot{\mathbf{x}}_{t+h}|\mathbf{x}_t, \dot{\mathbf{x}}_t, U) \sim \mathcal{N}\left(\dot{\mathbf{x}}_t + h f(\mathbf{x}_t), h g^2(\mathbf{x}_t)\right).$$

In the tests in section D.6 this scheme tends to overestimate the variance of the particle's position $\mathbf{x}_{t+h}$. A simple modification of the scheme that can still be easily evaluated for Langevin systems is the *semi-implicit midpoint Euler* scheme that modifies the position update step in equation (D.6) to be

$$\mathbf{x}_{t+h} = \mathbf{x}_t + 1/2(\dot{\mathbf{x}}_t + \dot{\mathbf{x}}_{t+h})h.$$

In the tests in section D.6, this scheme gave good estimates of the state variance over a single step. All these Euler schemes are consistent first order schemes [176].

D.2 Higher Order Scheme

Following the method of [323], a higher-order integration scheme for Langevin equations can be developed, although here the scheme is truncated at a lower level of accuracy than in [323] due to its application in system identification. The scheme requires higher order derivatives of the the f and g functions to be available. For clarity, the scheme is first developed in the single object case, and extended to the multi-object case in section D.2.1.

The system in (D.2)-(D.3) can be written in matrix-vector form as

$$dX = F(X)dt + G(X)dW_t \tag{D.7}$$

where

$$X(t) = \begin{bmatrix} \mathbf{x}(t) \\ \dot{\mathbf{x}}(t) \end{bmatrix}, \quad F(X) = \begin{bmatrix} \dot{\mathbf{x}} \\ f(\mathbf{x}) \end{bmatrix}, \quad G(X) = \begin{bmatrix} 0 \\ g(\mathbf{x}) \end{bmatrix}. \tag{D.8}$$

The update from time t to $t + h$ is given by

$$\delta X = \int_t^{t+h} dX = \int_t^{t+h} F(X)ds + \int_t^{t+h} G(X)dW_s. \tag{D.9}$$

To make further progress $F(X)$ is expanded *in space* (the X dimension) using a Taylor expansion. This is possible because $F(X)$ is assumed to be a smooth function in space. Expansion in the time dimension is *not* possible with a simple Taylor expansion because of the stochastic nature of $F(X)$ with respect to time. For simplicity of exposition $\mathbf{x}$ and $\dot{\mathbf{x}}$ will be considered as one-dimensional quantities x and $\dot{x}$ here; the following section details how to extend this to the multi-dimensional case. The standard Taylor series expansion is

$$\begin{aligned} F(X(t+r)) &= F(X(t)) + \left(\frac{\partial F}{\partial x}\right)_t \delta x + \left(\frac{\partial F}{\partial \dot{x}}\right)_t \delta \dot{x} \\ &\quad + \frac{1}{2}\left(\frac{\partial^2 F}{\partial x^2}\right)_t \delta x^2 + \left(\frac{\partial^2 F}{\partial x \partial \dot{x}}\right)_t \delta x \delta \dot{x} + \frac{1}{2}\left(\frac{\partial^2 F}{\partial \dot{x}^2}\right)_t \delta \dot{x}^2 + \ldots \end{aligned}$$

From (D.8) it can be seen that

$$\frac{\partial F}{\partial x} = \begin{bmatrix} 0 \\ \frac{\partial f}{\partial x} \end{bmatrix}, \quad \frac{\partial F}{\partial \dot{x}} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$\frac{\partial^2 F}{\partial x^2} = \begin{bmatrix} 0 \\ \frac{\partial^2 f}{\partial x^2} \end{bmatrix}, \quad \frac{\partial^2 F}{\partial x \partial \dot{x}} = \mathbf{0}, \quad \frac{\partial^2 F}{\partial \dot{x}^2} = \mathbf{0},$$

and so

$$F(X(t+r)) = F(X(t)) + \begin{bmatrix} \delta \dot{x} \\ \left(\frac{\partial f}{\partial x}\right)_t \delta x + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_t \delta x^2 + \ldots \end{bmatrix}. \tag{D.10}$$

To incorporate terms in f that depended on $\dot{x}$ (for example a resistance term) the second element of $\frac{\partial F}{\partial \dot{x}}$ would become $\frac{\partial f}{\partial \dot{x}}$ and so the expansion in the second component of (D.10) would be the full Taylor expansion of f with respect to both x and $\dot{x}$ (rather than just x). This is not considered here.

In order to carry out numerical integration it is necessary to express δX in terms of the timestep h. This can be done by substituting the expansion of F and a similar expansion of G into (D.9). In this work systems with constant noise are primarily of interest (at least in the first instance) and so henceforth it will be assumed that $g(x) = \sigma$; this assumption is not necessary, but relaxing it gives rise to a more complicated integrator. This gives an expression for δX

$$\delta X = \int_t^{t+h} F(X_t) + \begin{bmatrix} \delta\dot{x} \\ \left(\frac{\partial f}{\partial x}\right)_t \delta x + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_t \delta x^2 + \ldots \end{bmatrix} ds + \int_t^{t+h} \begin{bmatrix} 0 \\ \sigma \end{bmatrix} dW_s \quad \text{(D.11)}$$

This can be found in terms of h by noting that $\delta X = \begin{bmatrix} \delta x & \delta\dot{x} \end{bmatrix}'$ and using progressive approximations of δx and $\delta\dot{x}$ derived from equation (D.11) substituted back into (D.11) in order to derive increasingly high-order terms, as shown in [323] . In order to do this it is necessary to note that $\int_t^{t+r} dW_s$ is $O(r^{1/2})$ (since it is a Gaussian with mean 0 and *variance* r). Writing $\delta X_r^{(k)}$ to denote the terms of δX at time $t + r$ of order r^k, so that $\delta X_r = \sum_k \delta X_r^{(k)}$,

$$\delta X_r^{(1/2)} = \int_t^{t+r} \begin{bmatrix} 0 \\ \sigma \end{bmatrix} dW_s = \begin{bmatrix} 0 \\ \sigma Z_1(r) \end{bmatrix}$$

where $Z_1(r) = \int_t^{t+r} dW_s \sim \mathcal{N}(0, r)$. The first order terms are given by the first term in the ds integral

$$\delta X_r^{(1)} = \int_t^{t+r} F(X_t) ds = \begin{bmatrix} \dot{x}_t \\ f(x_t) \end{bmatrix} r.$$

The $O(h^{3/2})$ terms are the first to require substitution of the δX term back

into equation (D.11), since it is known that $\delta\dot{x}$ has a component of $O(r^{1/2})$ and thus the time integral of $\delta\dot{x}$ will lead to an $O(r^{3/2})$ term, so

$$\begin{aligned}\delta X_r^{(3/2)} &= \int_t^{t+r} \begin{bmatrix} \delta\dot{x}_s^{(1/2)} \\ 0 \end{bmatrix} ds \\ &= \int_t^{t+r} \begin{bmatrix} \sigma Z_1(s) \\ 0 \end{bmatrix} ds \\ &= \begin{bmatrix} \sigma Z_2(r) \\ 0 \end{bmatrix},\end{aligned}$$

where $Z_2(r) = \int_t^{t+r} Z_1(s)ds \sim \mathcal{N}(0, \frac{1}{3}r^3)$, and $\text{cov}\,(Z_1(r), Z_2(r)) = \frac{1}{2}r^2$. In order to get the next term $\delta X^{(2)}$, the the time integrals of the $O(r)$ components of δX need to be considered so that

$$\begin{aligned}\delta X_r^{(2)} &= \int_t^{t+r} \begin{bmatrix} \delta\dot{x}_s^{(1)} \\ \left(\frac{\partial f}{\partial x}\right)_t \delta x_s^{(1)} \end{bmatrix} ds \\ &= \int_t^{t+r} \begin{bmatrix} f(x_t)s \\ \left(\frac{\partial f}{\partial x}\right)_t \dot{x}_t s \end{bmatrix} ds \\ &= \begin{bmatrix} \frac{1}{2}r^2 f(x_t) \\ \frac{1}{2}r^2 \left(\frac{\partial f}{\partial x}\right)_t \dot{x}_t \end{bmatrix}.\end{aligned}$$

The next term in the expansion $\delta X^{(5/2)}$ is found by considering the time integral of any $O(r^{3/2})$ terms.

$$\begin{aligned}\delta X_r^{(5/2)} &= \int_t^{t+r} \begin{bmatrix} \delta\dot{x}_s^{(3/2)} \\ \left(\frac{\partial f}{\partial x}\right)_t \delta x_s^{(3/2)} \end{bmatrix} ds \\ &= \int_t^{t+r} \begin{bmatrix} 0 \\ \left(\frac{\partial f}{\partial x}\right)_t \sigma Z_2(s) \end{bmatrix} ds \\ &= \begin{bmatrix} 0 \\ \left(\frac{\partial f}{\partial x}\right)_t \sigma Z_3(r) \end{bmatrix}\end{aligned}$$

where $Z_3(r) = \int_t^{t+r} Z_2(s)ds \sim \mathcal{N}\left(0, \frac{r^5}{20}\right)$, with $\mathrm{cov}(Z_1(r), Z_3(r)) = \frac{1}{6}r^3$ and $\mathrm{cov}(Z_2(r), Z_3(r)) = \frac{1}{8}r^4$.

The $\delta X^{(3)}$ term is found by considering the time integral of any $O(r^2)$ terms, but these include $\delta\dot{x}$ terms up to $O(r^2)$ and δx terms up to $O(r)$ because δx^2 appears in the original expansion.

$$\begin{aligned}
\delta X_r^{(3)} &= \int_t^{t+r} \begin{bmatrix} \delta\dot{x}_s^{(2)} \\ \left(\frac{\partial f}{\partial x}\right)_t \delta x_s^{(2)} + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_t \left(\delta x_s^{(1)}\right)^2 \end{bmatrix} ds \\
&= \int_t^{t+r} \begin{bmatrix} \frac{1}{2}s^2\left(\frac{\partial f}{\partial x}\right)_t \dot{x}_t \\ \frac{1}{2}s^2 f(x_t)\left(\frac{\partial f}{\partial x}\right)_t + \frac{1}{2}\left(\frac{\partial^2 f}{\partial x^2}\right)_t (\dot{x}_t s)^2 \end{bmatrix} ds \\
&= \begin{bmatrix} \frac{1}{6}r^3\left(\frac{\partial f}{\partial x}\right)_t \dot{x}_t \\ \frac{1}{6}r^3 f(x_t)\left(\frac{\partial f}{\partial x}\right)_t + \frac{1}{6}r^3\left(\frac{\partial^2 f}{\partial x^2}\right)_t (\dot{x}_t)^2 \end{bmatrix}
\end{aligned}$$

Here, $\delta\dot{x}_r^{(3)}$ includes the first occurrence of the second partial derivative of f. For Gaussian process approximations of f this is likely to be quite unstable, with substantial variance in its estimates. Furthermore, the $\delta x_r^{(3)}$ term contains the partial derivative of f, complicating the functional form of the relationship between x_t, $\dot{x}_t$ and $f(x_t)$. Therefore, this $\delta X^{(3)}$ term is the last that is considered here. More accurate integrators are possible by continuing this line of reasoning, although the expansion here also makes it clear that the $\delta X^{(4)}$ term will contain a non-Gaussian term, as it will involve the time integral of $\left(\delta x^{(3/2)}\right)^2$, which will be the time integral of the square of a Gaussian.

The following numerical integrator, therefore, is suited to the inference task:

$$\delta X = \begin{bmatrix} \dot{x}_t h + \sigma Z_2 + \frac{1}{2}f(x_t)h^2 \\ \sigma Z_1 + f(x_t)h + \frac{1}{2}\left(\frac{\partial f}{\partial x}\right)_t \dot{x}_t h^2 + \sigma\left(\frac{\partial f}{\partial x}\right)_t Z_3 \end{bmatrix} + O(h^3).$$

with

$$\begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} h & h^2/2 & h^3/6 \\ h^2/2 & h^3/3 & h^4/8 \\ h^3/6 & h^4/8 & h^5/20 \end{bmatrix}\right)$$

D.2.1 Multivariate Scheme

The integrator above can be generalized fairly easily to multivariate $\mathbf{x}$ and $\dot{\mathbf{x}}$ (at least for low-order terms). For the n dimensional case (with constant noise)

$$\delta\mathbf{X} = \int_t^{t+h} F(\mathbf{X})ds + \int_t^{t+h} \begin{bmatrix} 0 \\ B \end{bmatrix} d\mathbf{W}_s$$

where

$$\delta\mathbf{X} = \begin{bmatrix} \delta x_1 \\ \dots \\ \delta x_n \\ \delta \dot{x}_1 \\ \dots \\ \delta \dot{x}_n \end{bmatrix}, \quad F(\mathbf{X}) = \begin{bmatrix} \dot{x}_1 \\ \dots \\ \dot{x}_n \\ f_1(\mathbf{x}) \\ \dots \\ f_n(\mathbf{x}) \end{bmatrix},$$

and B an $n \times n$ matrix giving the Cholesky decomposition of the noise covariance in the evolution of $\dot{\mathbf{x}}$. In this case $F(\mathbf{X})$ can be expanded using the multivariable Taylor expansion

$$\begin{aligned} F(\mathbf{X}(t+r)) &= F(\mathbf{X}(t)) + \sum_{i=1}^{n} \frac{\partial F}{\partial x_i}\delta x_i + \sum_{i=1}^{n} \frac{\partial F}{\partial \dot{x}_i}\delta \dot{x}_i \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 F}{\partial x_i x_j}\delta x_i \delta x_j + \text{H.O.T.} \end{aligned}$$

Note that here none of the $O(\delta x^2)$ terms involving derivatives of $\dot{x}_i$ are present because

$$\frac{\partial F}{\partial x_i} = \begin{bmatrix} 0 \\ \dots \\ 0 \\ \frac{\partial f_1}{\partial x_i} \\ \dots \\ \frac{\partial f_n}{\partial x_i} \end{bmatrix}, \quad \frac{\partial F}{\partial \dot{x}_i} = \begin{bmatrix} \delta_{i=1} \\ \dots \\ \delta_{i=n} \\ 0 \\ \dots \\ 0 \end{bmatrix},$$

where f_i is the i^{th} dimension of f function and $\delta_{i=j}$ is 1 when $i = j$ and 0 otherwise. Therefore, the second partial derivatives involving $\dot{x}_i$ are all zero. This means that

$$F(\mathbf{X}(t+r)) = F(\mathbf{X}(t)) + r \begin{bmatrix} I\delta\dot{\mathbf{x}} \\ J_t\delta\mathbf{x} + O(\delta x^2) \end{bmatrix}$$

where J_t is the Jacobian of f at time t. Therefore the same logic as in the univariate case can be followed in order to arrive at the numerical integrator

$$\delta\mathbf{X} = \begin{bmatrix} h\dot{\mathbf{x}}_t + BZ_2 + \frac{1}{2}h^2 f(\mathbf{x}_t) \\ BZ_1 + hf(\mathbf{x}_t) + \frac{1}{2}h^2 J_t\dot{\mathbf{x}}_t + J_t BZ_3 \end{bmatrix} + O(h^3) \tag{D.12}$$

where Z_1, Z_2 and Z_3 are multivariate (dimension n) Gaussian distributed random variables with zero mean and covariance given by

$$\operatorname{cov}\left(\begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix}\right) = \begin{bmatrix} hI_n & h^2/2I_n & h^3/6I_n \\ h^2/2I_n & h^3/3I_n & h^4/8I_n \\ h^3/6I_n & h^4/8I_n & h^5/20I_n \end{bmatrix}, \tag{D.13}$$

since the components of, say, Z_1 are independent of each other but correlated with the same component of Z_2 and Z_3.

In general the integrator in equation (D.12) cannot be inverted to find

both $f(\mathbf{x}_t)$ and J_t, since it provides $2n$ equations, but $f(x_t)$ and J_t contain $n(n+1)$ unknown quantities, so this system is only fully determined when $n = 1$. However, the first (position) part of the integrator can be inverted to obtain a noisy observation of $f(\mathbf{x}_t)$, since it only contains the n unknown components of $f(\mathbf{x}_t)$.

D.3 Transition Densities

Since these numerical integration schemes are to be used in filtering, it is necessary to evaluate their one-period transition densities, i.e. the (approximate) conditional distribution of X_{t+h} given X_t given by these integrators.

Conditioning additionally on the function f (and its Jacobian) equation (D.12) along with the distribution of the Z random variables in equation (D.13) can be used to get the transition density

$$p\left(\begin{bmatrix}\mathbf{x}_{t+h}\\ \dot{\mathbf{x}}_{t+h}\end{bmatrix} \mid X_t, f(\mathbf{x}_t), J_t\right) \sim \mathcal{N}\left(\mu_{t+h|t}, \Sigma_{t+h|t}\right) \tag{D.14}$$

with

$$\begin{aligned}\mu_{t+h|t} &= \begin{bmatrix}\mathbf{x}_t + h\dot{\mathbf{x}}_t + \frac{h^2}{2}f(\mathbf{x}_t)\\ \dot{\mathbf{x}}_t + hf(\mathbf{x}_t) + \frac{h^2}{2}J_t\dot{\mathbf{x}}_t\end{bmatrix}\\ \Sigma_{t+h|t} &= \begin{bmatrix}\frac{h^3}{3}\Sigma & \frac{h^2}{2}\Sigma + \frac{h^4}{8}\Sigma J_t^T\\ \frac{h^2}{2}\Sigma + \frac{h^4}{8}\Sigma J_t^T & h\Sigma + \frac{h^3}{3}\Sigma J_t^T + \frac{h^5}{20}J_t\Sigma J_t^T\end{bmatrix}\end{aligned}$$

where $\Sigma = BB^T$, the covariance of the noise in the original SDE.

This distribution is useful if a direct representation of the function f is known, for example in a particle filter in which each particle contains a sample of f.

In the work in chapter 5, however, the function f is treated as unknown and is modelled via a Gaussian process assumption. This leads to probabilistic estimates of $f(\mathbf{x}_t)$ and J_t, with these having a joint Gaussian dis-

tribution as described in section 5.2.2. Treating the f function as a random variable does not affect the validity of the numerical scheme in the cases of interest here. It is assumed that f exists and is smooth, so that the use of its space derivatives in deriving the numerical scheme remains valid. Having components of its realization at various points as (Gaussian) random variables simply models the fact that its value is not precisely known there. If the estimate of f is derived from previous observations then the uncertainty in that estimate is due to the state process noise up to the time of those observation, and the observation noise when making them. For the filtering models that are considered here, this noise is independent from future process and observation noise (and thus the Z variables). None of the expansions or integrals in section D.2 and D.2.1 are affected by f or its derivatives being random variables.

Treating $f(\mathbf{x}_t)$ and J_t in the integration scheme in equation (D.12) as random is troublesome since the term $J_t B Z_3$ occurs in the update for the $\dot{\mathbf{x}}$ component of the state. This term involves the sum of products of independent (non-zero mean) Gaussians (from J_t) and zero-mean Gaussians (from BZ_3) and will have a distribution something like a sum of product normal distributions, which will likely be distinctly non-Gaussian and its density will be difficult to calculate. Even if this could be found, the density of the sum of this with another Gaussain term (from the preceding terms) would be required. Perhaps progress could be made by making a Gaussian approximation to this distribution through, for example, moment matching, although it is not clear how good this would be and it is not pursued further here.

A simple solution is to disregard this term from the numerical integration, which leaves the scheme as $O(h^2)$ overall (although it remains $O(h^{5/2})$ for the position elements of the process) but has the significant advantage that all remaining terms are Gaussian. Doing this, we get the state update

equation

$$\begin{bmatrix} \mathbf{x}_{t+h} \\ \dot{\mathbf{x}}_{t+h} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_t \\ \dot{\mathbf{x}}_t \end{bmatrix} + \begin{bmatrix} h\dot{\mathbf{x}}_t + BZ_2 + \frac{1}{2}h^2 f(\mathbf{x}_t) \\ BZ_1 + hf(\mathbf{x}_t) + \frac{1}{2}h^2 J_t \dot{\mathbf{x}}_t \end{bmatrix} + \begin{bmatrix} O(h^3) \\ O(h^{5/2}) \end{bmatrix}. \quad \text{(D.15)}$$

Using the fact that the noise variables Z are independent of the uncertainty in the Gaussian process, and thus have zero covariance with the random variables arising from that, the state transition density is given by

$$p\left(\begin{bmatrix} \mathbf{x}_{t+h} \\ \dot{\mathbf{x}}_{t+h} \end{bmatrix} \mid X_{1:t} \right) \sim \mathcal{N}\left(\mu^*_{t+h|t}, \Sigma^*_{t+h|t} \right) \quad \text{(D.16)}$$

with

$$\begin{aligned} \mu^*_{t+h|t} &= \begin{bmatrix} \mathbf{x}_t + h\dot{\mathbf{x}}_t + \frac{h^2}{2}\bar{f}(\mathbf{x}_t) \\ \dot{\mathbf{x}}_t + h\bar{f}(\mathbf{x}_t) + \frac{h^2}{2}\bar{J}_t\dot{\mathbf{x}}_t \end{bmatrix} \\ \Sigma^*_{t+h|t} &= \begin{bmatrix} \operatorname{cov}(\mathbf{x}_{t+h}) & \operatorname{cov}(\mathbf{x}_{t+h}, \dot{\mathbf{x}}_{t+h}) \\ \operatorname{cov}(\mathbf{x}_{t+h}, \dot{\mathbf{x}}_{t+h}) & \operatorname{cov}(\dot{\mathbf{x}}_{t+h}) \end{bmatrix} \\ \operatorname{cov}(\mathbf{x}_{t+h}) &= \frac{h^3}{3}\Sigma + \frac{h^4}{4}\operatorname{cov}(f(\mathbf{x}_t), f(\mathbf{x}_t)) \\ \operatorname{cov}(\mathbf{x}_{t+h}, \dot{\mathbf{x}}_{t+h}) &= \frac{h^2}{2}\Sigma + \frac{h^3}{2}\operatorname{cov}(f(\mathbf{x}_t)) + \frac{h^4}{4}\operatorname{cov}(f(\mathbf{x}_t), J_t\dot{\mathbf{x}}_t) \\ \operatorname{cov}(\dot{\mathbf{x}}_{t+h}) &= h\Sigma + h^2\operatorname{cov}(f(\mathbf{x}_t)) + h^3\operatorname{cov}(f(\mathbf{x}_t), J_t\dot{\mathbf{x}}_t) \\ &\quad + \frac{h^4}{4}\operatorname{cov}(J_t\dot{\mathbf{x}}_t). \end{aligned}$$

In the above $\operatorname{cov}(\mathbf{y})$ refers to the covariance of the $\mathbf{y}$ vector with itself and $\bar{f}(\mathbf{x}_t)$ and $\bar{J}_t$ refer to the mean vector and matrix of the random vector $f(\mathbf{x}_t)$ and random matrix J_t, respectively.

In the case of a Gaussian process prior being applied to f the covariances are given by the mean of the process and its first derivative. The elements of $\operatorname{cov}(f(\mathbf{x}_t))$ are given by the covariance of the Gaussian process given by equation (5.7). The term $\operatorname{cov}(f(\mathbf{x}_t), J_t\dot{\mathbf{x}}_t)$ can be related to terms known from

the Gaussian process by

$$\begin{aligned} \mathrm{cov}(f(\mathbf{x}_t), J_t\mathbf{x}_t) &= \mathrm{cov}(f(\mathbf{x}_t), \dot{\mathbf{x}}_1 J_{\cdot 1} + \dot{\mathbf{x}}_2 J_{\cdot 2} + ...) \\ &= \sum_i \dot{\mathbf{x}}_i \mathrm{cov}(f(x_t), J_{\cdot i}) \end{aligned}$$

where $\dot{\mathbf{x}}_i$ is the i^{th} component of $\dot{\mathbf{x}}_t$ and $J_{\cdot i}$ is the i^{th} column of matrix J_t and the elements of $\mathrm{cov}(f(x_t), J_{\cdot i})$ are given by the covariance between the Gaussian process and its first derivatives from equation (5.10). Similarly, $\mathrm{cov}(J_t\dot{\mathbf{x}}_t)$ can be written in terms of known covariances from the Gaussian process since

$$\mathrm{cov}(J_t\dot{\mathbf{x}}_t) = \mathrm{cov}(\dot{\mathbf{x}}_1 J_{\cdot 1} + \dot{\mathbf{x}}_2 J_{\cdot 2} + ...) \quad \text{(D.17)}$$

$$= \sum_i \sum_j \dot{\mathbf{x}}_i \dot{\mathbf{x}}_j \mathrm{cov}(J_{\cdot i}, J_{\cdot j}) \quad \text{(D.18)}$$

where the elements of $\mathrm{cov}(J_{\cdot i}, J_{\cdot j})$ are given by the covariance between first derivatives of f from equation (5.11). In the case of the SLAM problem in chapter 5 the f function is taken to be the gradient of a Gaussian process and so further derivatives are required; these are given in section 5.2.2.

D.4 Trade-off Between Particle Filter Types

The difficulty in evaluating the state transition density in cases such as that above and in multi-step schemes (see section D.5, below) means that there is a trade-off between simulation-only bootstrap particle filters (and methods based on them) and other types of particle filter that require evaluation of the transition density.

Simulation-only filters can only propose from the transition distribution or distributions that can be related to this in a direct and simple way, such as one with a constant multiple of its variance. However, they are able to make use of more accurate numerical integration schemes that can be simulated from, but for which the transition density is intractable. These

schemes might offer better accuracy when the dynamical model is highly non-linear (so that simple integration schemes perform badly) with a reasonable level of state transition noise and not particularly informative observations. This is because, in this case, the distribution $p(x_t \mid x_{t-1})$ from which samples are drawn might be a better approximation of the posterior $p(x_t \mid x_{t-1}, y_t)$ (the ideal importance density) than a proposal that can consider the observation y_t but can only use an inaccurate numerical scheme to approximate $p(x_t \mid x_{t-1})$. On the other hand, if there is little noise in the dynamical model or if observations are very informative, the simulated trajectories are likely to be densely concentrated in areas that do not coincide well with the areas of high posterior density and the bootstap filter is likely to give much worse estimates than an approximately adapted filter with a less accurate integration scheme.

The appropriate choice of filter and numerical integration scheme is, unfortunately, therefore somewhat application dependent.

D.5 Multi-Step Schemes For Inference

Multi-step schemes using n integration steps of length $\frac{h}{n}$ in place of a single step of length h are an obvious way of reducing integration error. Such schemes are well suited to simulation at arbitrary accuracy, and any single step scheme can be used to make each step. However, in general the transition densities of such schemes will be intractable because the random variables generated in each step will undergo numerous non-linear mappings in subsequent steps. This limits the usefulness of these simple multi-step schemes for inference in all but simulation-only methods.

However, multi-step schemes can be developed for any single-step scheme in which the transition density is tractable using a *Bayesian imputation* approach [91; 326; 327; 58]. The basic idea of this is to set up a grid of variables $X_t, X_{t+\frac{h}{n}}, X_{t+\frac{2h}{n}}, ..., X_{t+h}$ at times between observations (here taken to be at t and $t+h$). The corresponding state space model for such a scheme is illus-

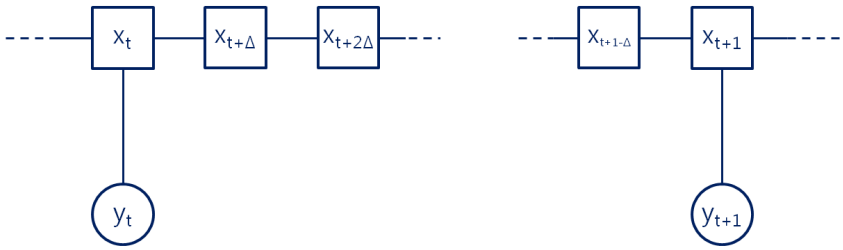


Figure D.1: State space model corresponding to the use of Bayesian imputation for numerical integration

trated in figure D.1. The transition density between these variables is then given by a numerical integration scheme with tractable likelihood. For the Gaussian diffusion systems considered here, this results in the transition density

$$p\left(X_{t+(k+1)\Delta} \mid X_{t+k\Delta}\right) = \mathcal{N}\left(X_{t+(k+1)\Delta}; \alpha\left(X_{t+k\Delta}, \Delta\right), \beta\left(X_{t+k\Delta}, \Delta\right)\right)$$

where $\Delta = h/n$, with the functions α and β given by the integration scheme. For example, for the Euler-Maruyama scheme for the system in equation (D.7)

$$\begin{aligned} \alpha\left(X_{t+k\Delta}, \Delta\right) &= F\left(X_{t+k\Delta}\right)\Delta \\ \beta\left(X_{t+k\Delta}, \Delta\right) &= \left(G(X_{t+k\Delta})\right)^2 \Delta. \end{aligned}$$

These intermediate variables can then be inferred using standard methods such as MCMC. Within the particle filter, the entire set of variables $\{X_{t+\frac{h}{n}}, X_{t+\frac{2h}{n}}, ..., X_{t+h}\}$ can be defined as the filter state at time $t + h$, and this can be proposed through recursive simulation and weighted in the usual way as in [58], for example. To approximately adapt the proposal to the next observation, Brownian bridge proposals are necessary (see e.g. chapter 6) to sample intermediate states.

An alternative method is to simulate from an accurate multi-step numerical scheme and then to use the samples generated to infer the first two

moments of the true distribution, which is known to be Gaussian in the Gaussian diffusion case. The unscented transform of [26] offers a possible efficient way to do this, although all such methods will require numerous evaluation of the SDEs' non-linear governing functions.

Such numerical schemes are computationally expensive, but offer a way of improving integration accuracy when single step methods with tractable likelihoods are insufficient and simulation-only methods do not work well. It might also be possible to use adaptive schemes to estimate the number of intermediate states necessary for the required levels of integration accuracy.

D.6 Comparison of Numerical Schemes

The Euler and higher-order methods shown in this appendix have been compared through numerical integration of the linear SDE system

$$\begin{bmatrix} dx \\ d\dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\lambda & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} dt + \begin{bmatrix} 0 \\ \sigma \end{bmatrix} dW_t.$$

This one dimensional system is a simple linear stochastic oscillator with a mean reversion coefficient of $-\lambda$ and can be analytically solved as shown in chapter 3. This can be viewed as the one dimensional motion of a particle inside a parabolic potential field $U = \frac{1}{2}\lambda^2$, subject to noise.

In order to test the methods a set of experiments were conducted using the parameters $\lambda = 10$ and $\sigma = 1$, and simulating the true system from $t = 0$ to $t = 2$ with the initial values $x = 0$ and $\dot{x} = 1$. Following this, numerical integration from $t = 2$ to $t = 3$ was performed using each method with a time step h of 0.1. A set of 20 experiments were conducted for each integration method, each with a different simulated path from $t = 0$ to $t = 2$. The integration methods were tested by generating 10,000 forward simulation paths and comparing the mean and standard deviation of these to the true values calculated from the analytic solution. Figures D.2 and D.3 show the results of one of these tests for multiple and single periods,

Method	error in mean	error in std. dev.
Position (x)		
Forward Euler	1.16 (0.618)	0.306 (0.00778)
Semi implicit Euler	0.271 (0.201)	0.124 (0.00932)
Semi implicit midpoint Euler	0.539 (0.331)	0.147 (0.0125)
Higher order (no Jacobian terms)	0.535 (0.338)	0.153 (0.0112)
Higher order (no Z_3 term)	0.107 (0.0572)	0.0331 (0.0101)
Higher order	0.107 (0.0572)	0.0136 (0.00555)
Velocity ($\dot{x}$)		
Forward Euler	3.71 (2.04)	0.757 (0.0272)
Semi implicit Euler	1.47 (0.881)	0.0732 (0.0237)
Semi implicit midpoint Euler	1.98 (1.01)	0.388 (0.0232)
Higher order (no Jacobian terms)	1.96 (1.02)	0.404 (0.0287)
Higher order (no Z_3 term)	0.341 (0.178)	0.1 (0.0254)
Higher order	0.34 (0.178)	0.041 (0.0148)

Table D.1: Sum (standard deviation) of absolute error over ten integration steps (from $t = 2.1$ to $t = 3$) in the estimate of the mean and of the standard deviation of position x and velocity $\dot{x}$ for a range of integration methods

respectively. The coloured lines shown are those from different integration methods, showing the mean and three standard deviations envelope for each method, generated by simulating 10,000 paths using each integrator.

The Euler methods tested were the forward, semi-implicit and semi-implicit midpoint Euler methods described in section D.1. The higher order methods tested were the one given in equation (D.11), and two variants, one without any terms involving the Jacobian and the other without the term involving Z_3, in order to see how well these simplified methods work. This led to the results in table D.1 giving the sum of *absolute errors* (not mean squared errors to avoid later errors completely dominating the calculations) between true and simulated means and standard deviations for both x and $\dot{x}$ over the ten integration steps from $t = 2.1$ to $t = 3$. Since one step ahead integration is of particular interest the same results were collected for the 1-step ahead point $t = 2.1$ and are given in table D.2.

From these results it can be seen that higher order methods work much

Method	error in mean	error in std. dev.
Position (x)		
Forward Euler	0.0126 (0.0105)	0.0181 (6.62e-014)
Semi implicit Euler	0.0129 (0.00948)	0.0136 (0.000197)
Midpoint implicit Euler	0.00259 (0.00153)	0.00227 (0.000108)
Higher order (no Jacobian terms)	0.00258 (0.00151)	0.000205 (0.000125)
Higher order (no Z_3 term)	0.00258 (0.00151)	0.000205 (0.000125)
Higher order	0.00258 (0.00151)	0.000205 (0.000125)
Velocity ($\dot{x}$)		
Forward Euler	0.0757 (0.0444)	0.0053 (0.00235)
Semi implicit Euler	0.0741 (0.0454)	0.00552 (0.00197)
Midpoint implicit Euler	0.0765 (0.0454)	0.00523 (0.00203)
Higher order (no Jacobian terms)	0.0756 (0.0453)	0.00503 (0.00258)
Higher order (no Z_3 term)	0.00527 (0.00426)	0.00503 (0.00258)
Higher order	0.00523 (0.00424)	0.00209 (0.00136)

Table D.2: Sum (standard deviation) of absolute error over a single integration step in the estimate of the mean and of the standard deviation of position x and velocity $\dot{x}$ for a range of integration methods

better than the Euler methods over one or several periods. Even the simplest higher order method involving no Jacobians is as good as the best Euler scheme, the semi-implicit midpoint Euler. Removal of the term including Z_3 degrades the estimation of the standard deviation for the velocity compared to the full scheme, but otherwise does not significantly affect performance over one period.

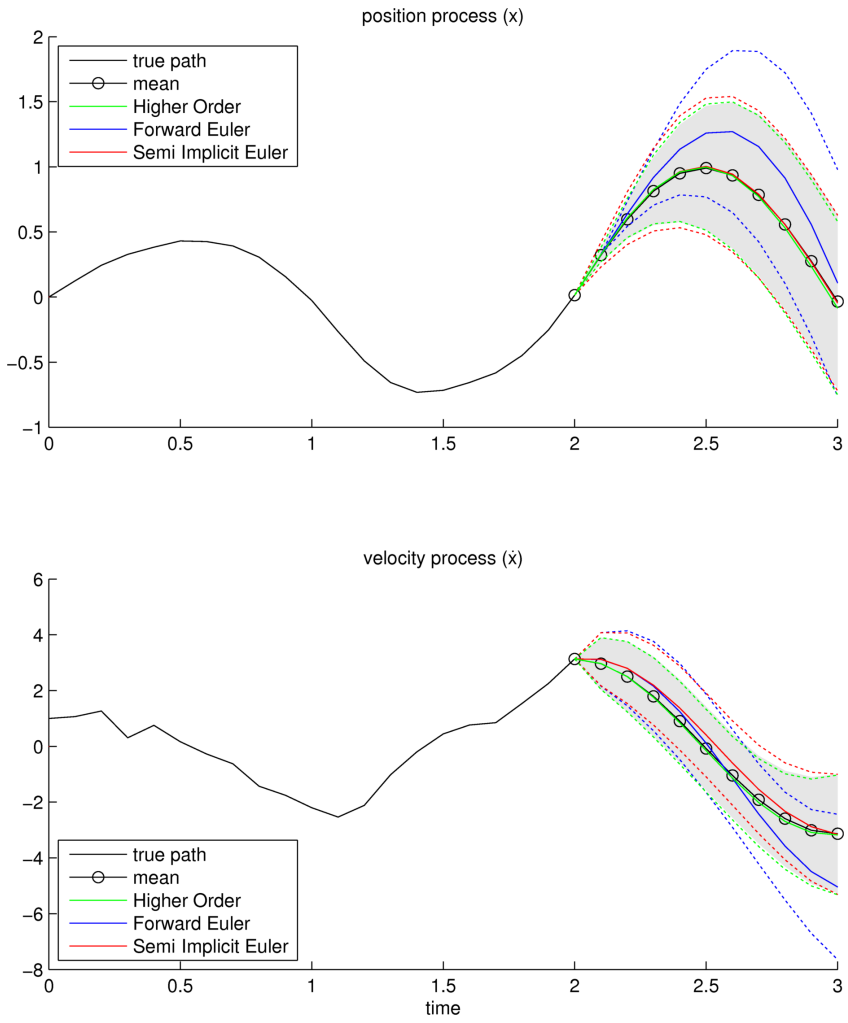


Figure D.2: Comparison of multi-period integration by three methods, forward Euler (blue), semi implicit Euler (red) and the higher order method described in the text (green). Solid lines show mean of 10,000 simulated paths and dotted lines show three standard deviations from the mean. The true mean and three standard deviations are shown by the black circles and grey shading, respectively. The upper graph shows the position x results and the lower graph shows the velocity $\dot{x}$ results. The preceding simulation is shown by a solid black line

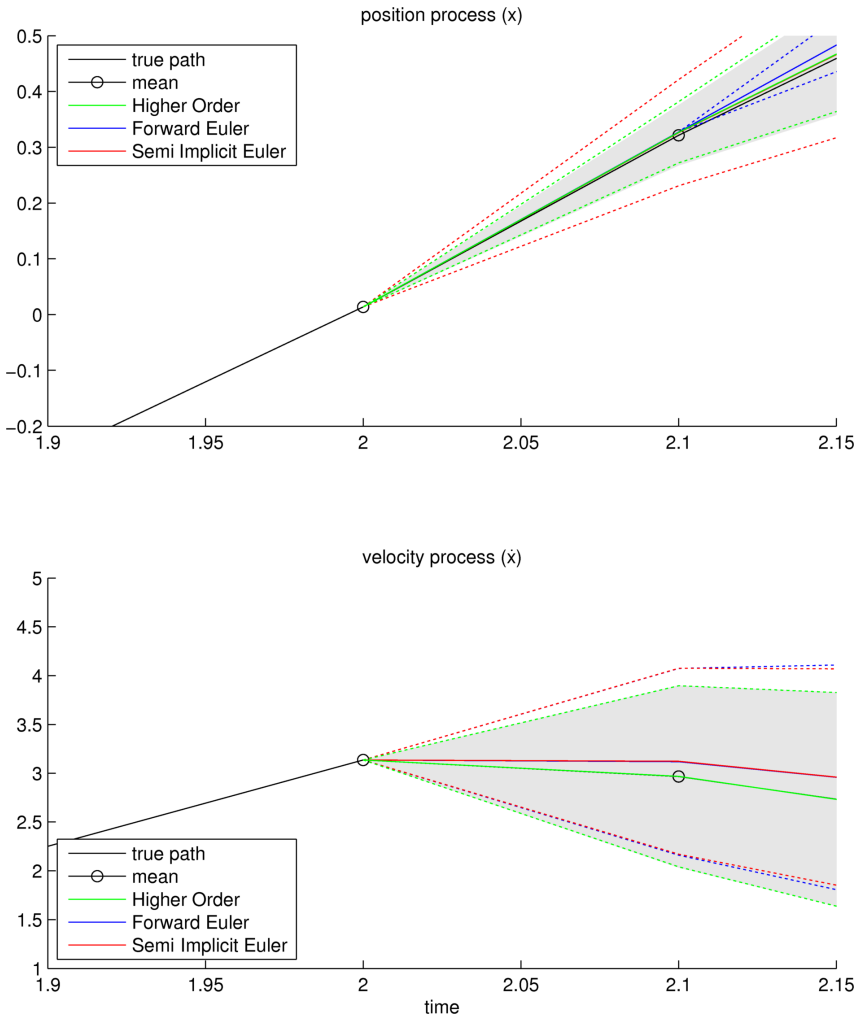


Figure D.3: Detail from figure D.2 above, showing single period integration by each of the three methods for the first period

Bibliography

[1] J. Murphy and S. Godsill, "Joint Bayesian removal of impulse and background noise," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, pp. 261–264, IEEE, 2011.

[2] H. Christensen, J. Murphy, and S. Godsill, "Forecasting high-frequency futures returns using online Langevin dynamics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 4, pp. 366–380, 2012.

[3] J. Murphy and S. Godsill, "Simultaneous localization and mapping for non-parametric potential field environments," in *Workshop on Sensor Data Fusion: Trends, Solutions, Applications (SDF2012)*, pp. 1–6, IEEE, 2012.

[4] J. Murphy and S. Godsill, "Structure inference for networks with general non-parametric inter-object relationships," in *Proceedings of 15th International Conference on Information Fusion*, IEEE, July 2012.

[5] J. Berger, "The case for objective Bayesian analysis," *Bayesian Analysis*, vol. 1, no. 3, pp. 385–402, 2006.

[6] B. De Finetti, *Theory of Probability: A critical introductory treatment. Vol. 1*. Wiley, 1974.

[7] M. Goldstein, "Subjective Bayesian analysis: principles and practice," *Bayesian Analysis*, vol. 1, no. 3, pp. 403–420, 2006.

[8] R. T. Cox, "Probability, frequency and reasonable expectation," *American Journal of Physics*, vol. 14, p. 1, 1946.

[9] K. S. Van Horn, "Constructing a logic of plausible inference: a guide to Cox's theorem," *International Journal of Approximate Reasoning*, vol. 34, no. 1, pp. 3–24, 2003.

[10] I. Hacking, "Slightly more realistic personal probability," *Philosophy of Science*, vol. 34, pp. 311–325, December 1967.

[11] M. Colyvan, "Is probability the only coherent approach to uncertainty?," *Risk Analysis*, vol. 28, no. 3, pp. 645–652, 2008.

[12] T. Bayes and R. Price, "An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, FRS communicated by Mr. Price, in a letter to John Canton, AMFRS," *Philosophical Transactions (1683-1775)*, pp. 370–418, 1763.

[13] P.-S. Laplace and S. M. Stigler (trans.), "Mémoire sur la probabilité des cause par les événments (1774) (English translation)," *Statistical Sciences*, vol. 1, no. 3, pp. 359–378, 1986.

[14] G. Kitagawa, "The two-filter formula for smoothing and an implementation of the Gaussian-sum smoother," *Annals of the Institute of Statistical Mathematics*, vol. 46, no. 4, pp. 605–623, 1994.

[15] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state-space models," *Annals of the Institute of Statistical Mathematics*, vol. 62, no. 1, pp. 61–89, 2010.

[16] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[17] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[18] Y. Ho and R. Lee, "A Bayesian approach to problems in stochastic estimation and control," *IEEE Transactions on Automatic Control*, vol. 9, pp. 333–339, 1964.

[19] A. Harvey, *Forecasting, structual timeseries models and the Kalman filter*. Cambridge University Press, 1990.

[20] H. E. Rauch, C. Striebel, and F. Tung, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[21] M. Briers, A. Doucet, and S. Maskell, "Smoothing algorithms for state-space models," Tech. Rep. F-INFENG.TR. 498, Cambridge University CUED, 2004.

[22] G. J. Bierman, "Fixed interval smoothing with discrete measurements," *International Journal of Control*, vol. 18, no. 1, pp. 65–75, 1973.

[23] M. S. Grewal and A. P. Andrews, "Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]," *Control Systems, IEEE*, vol. 30, no. 3, pp. 69–78, 2010.

[24] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[25] M. Athans, R. Wishner, and A. Bertolini, "Suboptimal state estimation for continuous-time nonlinear systems from discrete noisy measurements," *IEEE Transactions on Automatic Control*, vol. 13, no. 5, pp. 504–514, 1968.

[26] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Proceedings of AeroSense '97*, pp. 182–193, International Society for Optics and Photonics, 1997.

[27] E. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pp. 153–158, IEEE, 2000.

[28] K. Ito and K. Xiong, "Gaussian filters for nonlinear filtering problems," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 910–927, 2000.

[29] E. A. Wan and R. Van Der Merwe, *The unscented Kalman filter (in Kalman filtering and neural networks)*, ch. 7, pp. 221–280. Wiley, 2001.

[30] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the American Control Conference*, vol. 6, pp. 4555–4559, IEEE, 2002.

[31] R. Van Der Merwe and E. A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001 (ICASSP'01)*, vol. 6, pp. 3461–3464, IEEE, 2001.

[32] S. Sarkka, "Unscented Rauch–Tung–Striebel smoother," *Automatic Control, IEEE Transactions on*, vol. 53, no. 3, pp. 845–849, 2008.

[33] C. Andrieu, A. Doucet, and R. Holenstein, "Particle Markov chain Monte Carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[34] P. Green, "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination," *Biometrika*, vol. 82, no. 4, p. 711, 1995.

[35] S. Godsill and J. Vermaak, "Variable rate particle filters for tracking applications," *Proceedings of 13th IEEE/SP Workshop on Statistical Signal Processing*, pp. 1280–1285, 2005.

[36] D. MacKay, *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[37] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, pp. 1–38, 1977.

[39] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[40] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[41] C. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, no. 1, pp. 95–103, 1983.

[42] R. Shumway and D. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[43] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[44] Z. Ghahramani and S. T. Roweis, "Learning nonlinear dynamical systems using an EM algorithm," *Advances in Neural Information Processing Systems*, pp. 431–437, 1999.

[45] A. Logothetis and V. Krishnamurthy, "Expectation maximization algorithms for MAP estimation of jump Markov linear systems," *IEEE Transactions on Signal Processing*, vol. 47, no. 8, pp. 2139–2156, 1999.

[46] M. J. Beal, *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

[47] M. Beal, F. Falciani, Z. Ghahramani, C. Rangel, and D. Wild, "A Bayesian approach to reconstructing genetic regulatory networks with hidden factors," *Bioinformatics*, vol. 21, no. 3, pp. 349–356, 2005.

[48] R. Mehra, "Approaches to adaptive filtering," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 693–698, 1972.

[49] D. Magill, "Optimal adaptive estimation of sampled stochastic processes," *IEEE Transactions on Automatic Control*, vol. 10, no. 4, pp. 434–439, 1965.

[50] R. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, 1970.

[51] X. R. Li and V. P. Jilkov, "A survey of maneuvering target tracking, part V: Multiple-model methods," in *Conference on Signal and Data Processing of Small Targets*, vol. 4473, pp. 559–581, 2003.

[52] H. A. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with Markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[53] X. R. Li and Y. Bar-Shalom, "A recursive multiple model approach to noise identification," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 3, pp. 671–684, 1994.

[54] S. Sarkka and A. Nummenmaa, "Recursive noise adaptive Kalman filtering by variational Bayesian approximations," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 596–600, 2009.

[55] J. N. Yang, S. Lin, H. Huang, and L. Zhou, "An adaptive extended Kalman filter for structural damage identification," *Structural Control and Health Monitoring*, vol. 13, no. 4, pp. 849–867, 2006.

[56] Q. Song and J.-D. Han, "An adaptive UKF algorithm for the state and parameter estimations of a mobile robot," *Acta Automatica Sinica*, vol. 34, no. 1, pp. 72–79, 2008.

[57] J. Liu and M. West, *Combined parameter and state estimation in simulation-based filtering (in Sequential Monte Carlo methods in practice)*, ch. 10, pp. 197–224. Springer, 2001.

[58] A. Golightly and D. J. Wilkinson, "Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo," *Interface Focus*, vol. 1, no. 6, pp. 807–820, 2011.

[59] A. Beskos, O. Papaspiliopoulos, G. O. Roberts, and P. Fearnhead, "Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes (with discussion)," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 333–382, 2006.

[60] A. Beskos, O. Papaspiliopoulos, and G. O. Roberts, "Retrospective exact simulation of diffusion sample paths with applications," *Bernoulli*, vol. 12, no. 6, pp. 1077–1098, 2006.

[61] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, pp. 107–113, IET, 1993.

[62] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate Bayesian computation in population genetics," *Genetics*, vol. 162, no. 4, pp. 2025–2035, 2002.

[63] P. Fearnhead and D. Prangle, "Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 74, no. 3, pp. 419–474, 2012.

[64] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré, "Markov chain Monte Carlo without likelihoods," *Proceedings of the National Academy of Sciences*, vol. 100, no. 26, pp. 15324–15328, 2003.

[65] K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François, "Approximate Bayesian computation (ABC) in practice," *Trends in Ecology & Evolution*, vol. 25, no. 7, pp. 410–418, 2010.

[66] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, p. 1087, 1953.

[67] W. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[68] A. Gelfand and A. Smith, "Sampling-based approaches to calculating marginal densities," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 398–409, 1990.

[69] W. Gilks, S. Richardson, and D. Spiegelhalter, eds., *Markov chain Monte Carlo in practice*. Chapman & Hall, 1996.

[70] L. Tierney, "Markov chains for exploring posterior distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.

[71] K. B. Athreya, H. Doss, and J. Sethuraman, "On the convergence of the markov chain simulation method," *The Annals of Statistics*, vol. 24, no. 1, pp. 69–100, 1996.

[72] G. Roberts and A. Smith, "Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms," *Stochastic Processes and their Applications*, vol. 49, no. 2, pp. 207–216, 1994.

[73] G. Roberts, *Strategies for improving MCMC (in Markov chain Monte Carlo in practice)*, ch. 3, pp. 45–58. Chapman & Hall, 1996.

[74] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.

[75] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks, "BUGS: Bayesian inference using Gibbs sampling, Version 0.30," *MRC Biostatistics Unit, Cambridge*, 1994.

[76] J. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Statistics and Computing*, vol. 6, no. 2, pp. 113–119, 1996.

[77] S. Brooks, "Markov chain Monte Carlo method and its application," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 47, no. 1, pp. 69–100, 1998.

[78] A. Gelfand and S. Sahu, "On Markov chain Monte Carlo acceleration," *Journal of Computational and Graphical Statistics*, vol. 3, no. 3, pp. 261–276, 1994.

[79] C. Andrieu and É. Moulines, "On the ergodicity properties of some adaptive MCMC algorithms," *The Annals of Applied Probability*, vol. 16, no. 3, pp. 1462–1505, 2006.

[80] G. O. Roberts and J. S. Rosenthal, "Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms," *Journal of Applied Probability*, pp. 458–475, 2007.

[81] Y. Atchadé and J. Rosenthal, "On adaptive Markov chain Monte Carlo algorithms," *Bernoulli*, vol. 11, no. 5, pp. 815–828, 2005.

[82] W. Gilks and G. Roberts, *Strategies for improving MCMC (in Markov Chain Monte Carlo in Practice)*, ch. 6, pp. 89–114. Chapman & Hall, 1996.

[83] G. Roberts and J. Rosenthal, "Examples of adaptive MCMC," *Journal of Computational and Graphical Statistics*, vol. 18, no. 2, pp. 349–367, 2009.

[84] J. S. Rosenthal, *Optimal proposal distributions and adaptive MCMC (in Handbook of Markov Chain Monte Carlo)*, ch. 4, pp. 93–112. Chapman & Hall, 2011.

[85] H. Haario, E. Saksman, and J. Tamminen, "An adaptive Metropolis algorithm," *Bernoulli*, vol. 7(2), no. 2, pp. 223–242, 2001.

[86] H. Haario, M. Laine, A. Mira, and E. Saksman, "DRAM: efficient adaptive MCMC," *Statistics and Computing*, vol. 16, no. 4, pp. 339–354, 2006.

[87] W. Gilks, G. Roberts, and S. Sahu, "Adaptive Markov chain Monte Carlo through regeneration," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1045–1054, 1998.

[88] Y. Atchadé, G. Fort, E. Moulines, and P. Priouret, *Inference and Learning in Dynamic Models (in Adaptive Markov chain Monte Carlo: theory and methods)*. Cambridge University Press, To Appear.

[89] H. Tian, T. Shen, B. Hao, Y. Hu, and N. Yang, "Image restoration based on adaptive MCMC particle filter," in *Proceedings of 2nd International Congress on Image and Signal Processing (CISP'09)*, pp. 1–5, IEEE, 2009.

[90] G. Roberts, A. Gelman, and W. Gilks, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *Annals of Applied Probability*, vol. 7, no. 1, pp. 110–120, 1997.

[91] G. Roberts and J. Rosenthal, "Optimal scaling for various Metropolis-Hastings algorithms," *Statistical Science*, vol. 16, no. 4, pp. 351–367, 2001.

[92] S. L. Alder, "Over-relaxation method for the Monte Carlo evaluation of the partition function for multiquadratic actions," *Physical Review D (Particles and Fields)*, vol. 23, no. 12, pp. 2901–2904, 1981.

[93] R. Neal, *Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation (in Learning in Graphical Models)*, pp. 205–225. Kluwer Academic Publishers, 1998.

[94] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid Monte Carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[95] R. Neal, "Markov chain Monte Carlo methods based on slicing the density function," *Technical Report 9722 Department of Statistics University of Toronto*, 1997.

[96] R. Neal, "Slice sampling," *Annals of Statistics*, vol. 31, no. 3, pp. 705–741, 2003.

[97] E. Marinari and G. Parisi, "Simulated tempering: a new Monte Carlo scheme," *EPL (Europhysics Letters)*, vol. 19, pp. 451–458, 1992.

[98] C. J. Geyer and E. A. Thompson, "Annealing Markov chain Monte Carlo with applications to ancestral inference," *Journal of the American Statistical Association*, vol. 90, no. 431, pp. 909–920, 1995.

[99] R. H. Swendsen and J.-S. Wang, "Replica Monte Carlo simulation of spin-glasses," *Physical Review Letters*, vol. 57, no. 21, pp. 2607–2609, 1986.

[100] C. J. Geyer, "Markov chain Monte Carlo maximum likelihood," in *Computing Science and statistics: proceedings of the 23rd symposium on the interface*, pp. 156–163, Interface Foundation, 1991.

[101] Y. Iba, "Extended ensemble Monte Carlo," *International Journal of Modern Physics C*, vol. 12, no. 5, pp. 623–656, 2001.

[102] Y. F. Atchadé, G. O. Roberts, and J. S. Rosenthal, "Towards optimal scaling of Metropolis-coupled Markov chain Monte Carlo," *Statistics and Computing*, vol. 21, no. 4, pp. 555–568, 2011.

[103] J. Propp and D. Wilson, "Exact sampling with coupled Markov chains and applications to statistical mechanics," *Random Structures and Algorithms*, vol. 9, no. 1-2, pp. 223–252, 1996.

[104] S. Brooks, Y. Fan, and J. Rosenthal, "Perfect forward simulation via simulated tempering," *Communications in Statistics-Simulation and Computation*, vol. 35, no. 3, pp. 683–713, 2006.

[105] P. J. Green and D. I. Hastie, "Reversible jump MCMC," *Genetics*, vol. 155, no. 3, pp. 1391–1403, 2009.

[106] M. A. Beaumont, "Estimation of population growth or decline in genetically monitored populations," *Genetics*, vol. 164, no. 3, pp. 1139–1160, 2003.

[107] C. Andrieu and G. O. Roberts, "The pseudo-marginal approach for efficient Monte Carlo computations," *The Annals of Statistics*, vol. 37, no. 2, pp. 697–725, 2009.

[108] D. J. Wilkinson, "The pseudo-marginal approach to "exact approximate" MCMC algorithms." Web (url: http://darrenjw.wordpress.com /2010/09/20/the-pseudo-marginal-approach-to-exact-approximate-mcmc-algorithms/), September 2010.

[109] M. Cowles and B. Carlin, "Markov chain Monte Carlo convergence diagnostics: a comparative review," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 883–904, 1996.

[110] S. Brooks and A. Gelman, "General methods for monitoring convergence of iterative simulations," *Journal of Computational and Graphical Statistics*, vol. 7, no. 4, pp. 434–455, 1998.

[111] K. L. Mengersen, C. P. Robert, and C. Guihenneuc-Jouyaux, "Mcmc convergence diagnostics: a "reviewww"," *Bayesian Statistics*, vol. 6, pp. 415–440, 1999.

[112] J. Nylander, J. Wilgenbusch, D. Warren, and D. Swofford, "AWTY(are we there yet?): a system for graphical exploration of MCMC convergence in Bayesian phylogenetics," *Bioinformatics*, vol. 24, no. 4, p. 581, 2008.

[113] J. Peltonen, J. Venna, and S. Kaski, "Visualizations for assessing convergence and mixing of Markov chain Monte Carlo simulations," *Computational Statistics & Data Analysis*, vol. 53, no. 12, pp. 4453–4470, 2009.

[114] M. Cowles, G. Roberts, and J. Rosenthal, "Possible biases induced by MCMC convergence diagnostics," *Journal of Statistical Computation and Simulation*, vol. 64, no. 1, pp. 87–104, 1999.

[115] J. Flegal and G. Jones, *Implementing MCMC: estimating with confidence (in Handbook of Markov chain Monte Carlo)*, ch. 7, pp. 175–198. Chapman & Hall, 2011.

[116] P. Del Moral, A. Doucet, and A. Jasra, "Sequential Monte Carlo samplers," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.

[117] R. Bucy and K. Senne, "Digital synthesis of non-linear filters," *Automatica*, vol. 7, no. 3, pp. 287–298, 1971.

[118] A. Wang and R. Klein, "Implementation of non-linear estimators using monospline," in *1976 IEEE Conference on Decision and Control including the 15th Symposium on Adaptive Processes*, vol. 15, 1976.

[119] S. Kramer and H. Sorenson, "Recursive Bayesian estimation using piece-wise constant approximations," *Automatica*, vol. 24, no. 6, pp. 789–801, 1988.

[120] G. Kitagawa, "Non-Gaussian state-space modeling of nonstationary time series," *Journal of the American Statistical Association*, vol. 82, no. 400, pp. 1032–1041, 1987.

[121] J. Carpenter, P. Clifford, and P. Fearnhead, "Improved particle filter for nonlinear problems," *IEE Proceedings-Radar, Sonar and Navigation*, vol. 146, no. 1, pp. 2–7, 1999.

[122] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. 1–25, 1996.

[123] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," *Handbook of Nonlinear Filtering*, vol. 12, pp. 656–704, 2009.

[124] P. Del Moral, *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer, 2004.

[125] M. K. Pitt, R. d. S. Silva, P. Giordani, and R. Kohn, "On some properties of Markov chain Monte Carlo simulation methods based on the particle filter," *Journal of Econometrics*, vol. 171, no. 2, pp. 134–151, 2012.

[126] A. Blake and M. Isard, "The condensation algorithm-conditional density propagation and applications to visual tracking," *Advances in Neural Information Processing Systems*, pp. 361–367, 1997.

[127] J. Liu and R. Chen, "Blind deconvolution via sequential imputations," *Journal of the American Statistical Association*, vol. 90, no. 430, 1995.

[128] A. Kong, J. Liu, and W. Wong, "Sequential imputations and Bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[129] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks," in *Proceedings of the 16th conference on Uncertainty in Artificial Intelligence*, pp. 176–183, 2000.

[130] O. Cappé, S. Godsill, and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 899–924, 2007.

[131] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 64–69, IEEE, 2005.

[132] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, 2002.

[133] P. Del Moral, "Non-linear filtering: interacting particle resolution," *Markov Processes and Related Fields*, vol. 2, no. 4, pp. 555–581, 1996.

[134] P. Del Moral and A. Guionnet, "Central limit theorem for nonlinear filtering and interacting particle systems," *Annals of Applied Probability*, vol. 9, no. 2, pp. 275–297, 1999.

[135] N. Chopin, "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference," *Annals of Statistics*, vol. 32, no. 6, pp. 2385–2411, 2004.

[136] S. K. Pang, J. Li, and S. J. Godsill, "Models and algorithms for detection and tracking of coordinated groups," in *Proceedings of IEEE Aerospace Conference*, pp. 1–17, IEEE, 2008.

[137] S. Godsill and T. Clapp, *Improvement strategies for Monte Carlo particle filters (in Sequential Monte Carlo Methods in Practice)*, ch. 7, pp. 139–158. Springer, 2001.

[138] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," *Advances in Neural Information Processing Systems*, pp. 584–590, 2001.

[139] M. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, pp. 590–599, 1999.

[140] W. Gilks and C. Berzuini, "Following a moving target–Monte Carlo inference for dynamic Bayesian models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 1, pp. 127–146, 2001.

[141] N. Polson, J. Stroud, and P. Müller, "Practical filtering with sequential parameter learning," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 2, pp. 413–428, 2008.

[142] Z. Khan, T. Balch, and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 11, pp. 1805–1819, 2005.

[143] S. N. MacEachern, M. Clyde, and J. S. Liu, "Sequential importance sampling for nonparametric Bayes models: The next generation," *Canadian Journal of Statistics*, vol. 27, no. 2, pp. 251–267, 1999.

[144] T. Schon, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, 2005.

[145] P. Fearnhead, D. Wyncoll, and J. Tawn, "A sequential smoothing algorithm with linear computational cost," *Biometrika*, vol. 97, no. 2, pp. 447–464, 2010.

[146] S. J. Godsill, A. Doucet, and M. West, "Monte Carlo smoothing for nonlinear time series," *Journal of the American Statistical Association*, vol. 99, no. 465, pp. 156–168, 2004.

[147] S. Godsill, A. Doucet, and M. West, "Maximum a posteriori sequence

estimation using Monte Carlo particle filters," *Annals of the Institute of Statistical Mathematics*, vol. 53, no. 1, pp. 82–96, 2001.

[148] M. Klaas, M. Briers, N. De Freitas, A. Doucet, S. Maskell, and D. Lang, "Fast particle smoothing: if I had a million particles," in *Proceedings of the 23rd International Conference on Machine Learning*, pp. 481–488, ACM, 2006.

[149] N. Kantas, A. Doucet, S. S. Singh, and J. M. Maciejowski, "An overview of sequential Monte Carlo methods for parameter estimation in general state-space models," in *Proceedings of 15th IFAC Symposium on System Identification*, vol. 15, 2009.

[150] G. Celeux and J. Diebolt, "The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem," *Computational Statistics Quarterly*, vol. 2, no. 1, pp. 73–82, 1985.

[151] P. Fearnhead, "Markov chain Monte Carlo, sufficient statistics, and particle filters," *Journal of Computational and Graphical Statistics*, vol. 11, no. 4, pp. 848–862, 2002.

[152] G. Storvik, "Particle filters for state-space models with the presence of unknown static parameters," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 281–289, 2002.

[153] A. Wills, T. B. Schön, and B. Ninness, "Parameter estimation for discrete-time nonlinear systems using EM," in *Proceedings of the 17th IFAC World Congress, Seoul, South Korea*, vol. 2, p. 93, 2008.

[154] C. Andrieu, A. Doucet, and V. B. Tadic, "On-line parameter estimation in general state-space models," in *Proceedings of 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05)*, pp. 332–337, IEEE, 2005.

[155] R. A. Fisher, "Theory of statistical estimation," in *Mathematical Pro-*

ceedings of the Cambridge Philosophical Society, vol. 22, pp. 700–725, Cambridge University Press, 1925.

[156] G. Poyiadjis, A. Doucet, and S. Singh, "Maximum likelihood parameter estimation in general state-space models using particle methods," *Proc of the American Stat. Assoc*, 2005.

[157] A. Doucet and V. Tadić, "Parameter estimation in general state-space models using particle methods," *Annals of the institute of Statistical Mathematics*, vol. 55, no. 2, pp. 409–422, 2003.

[158] C. Andrieu, A. Doucet, S. S. Singh, and V. B. Tadic, "Particle methods for change detection, system identification, and control," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 423–438, 2004.

[159] M. West, "Approximating posterior distributions by mixtures," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 55, no. 2, pp. 409–422, 1993.

[160] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[161] C. Andrieu and A. Doucet, "Online expectation-maximization type algorithms for parameter estimation in general state space models," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, vol. 6, pp. 69–72, IEEE, 2003.

[162] G. Poyiadjis, A. Doucet, and S. S. Singh, "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation," *Biometrika*, vol. 98, no. 1, pp. 65–80, 2011.

[163] P. Del Moral, A. Doucet, and S. Singh, "Forward smoothing using sequential Monte Carlo," *arXiv preprint arXiv:1012.5390*, 2010.

[164] J. Murphy, "Bayesian methods for high frequency financial time series analysis," *PhD First Year Report, Cambridge University Department of Engineering*, 2010.

[165] R. Merton, "Option pricing when underlying stock returns are discontinuous," *Journal of Financial Economics*, vol. 3, no. 1-2, pp. 125–144, 1976.

[166] H. Yang and L. Zhang, "Optimal investment for insurer with jump-diffusion risk process," *Insurance: Mathematics and Economics*, vol. 37, no. 3, pp. 615–634, 2005.

[167] C. Zhou, "A jump-diffusion approach to modeling credit risk and valuing defaultable securities," *Working Paper, Federal Reserve Board (available at SSRN 39800)*, vol. 97, no. 15, 1997.

[168] R. Weron, M. Bierbrauer, and S. Trück, "Modeling electricity prices: jump diffusion and regime switching," *Physica A: Statistical Mechanics and its Applications*, vol. 336, no. 1, pp. 39–48, 2004.

[169] T. Meyer-Brandis and P. Tankov, "Multi-factor jump-diffusion models of electricity prices," *International Journal of Theoretical and Applied Finance*, vol. 11, no. 5, pp. 503–528, 2008.

[170] A. Cartea and M. G. Figueroa, "Pricing in electricity markets: a mean reverting jump diffusion model with seasonality," *Applied Mathematical Finance*, vol. 12, no. 4, pp. 313–335, 2005.

[171] C. Chudley and R. Elliott, "Neutron scattering from a liquid on a jump diffusion model," *Proceedings of the Physical Society*, vol. 77, no. 2, p. 353, 1961.

[172] S. Godsill, "Particle filters for continuous-time jump models in tracking applications," *ESAIM: Proceedings*, vol. 19, pp. 39–52, 2007.

[173] S. Godsill, J. Vermaak, W. Ng, and J. Li, "Models and algorithms for tracking of maneuvering objects using variable rate particle filters," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 925–952, 2007.

[174] M. S. Johannes, N. G. Polson, and J. R. Stroud, "Optimal filtering of jump diffusions: Extracting latent states from asset prices," *Review of Financial Studies*, vol. 22, no. 7, pp. 2759–2799, 2009.

[175] S. Godsill and J. Vermaak, "Models and algorithms for tracking using trans-dimensional sequential monte carlo," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, vol. 3, pp. iii–976, IEEE, 2004.

[176] P. E. Kloeden and E. Platen, *Numerical solution of stochastic differential equations*. Springer Verlag, 1992.

[177] S. Särkkä, "Recursive Bayesian inference on stochastic differential equations," *Doctoral Dissertation, Helsinki University of Technology*, 2006.

[178] C. K. Carter and R. Kohn, "On Gibbs sampling for state space models," *Biometrika*, vol. 81, no. 3, pp. 541–553, 1994.

[179] S. Frühwirth-Schnatter, "Data augmentation and dynamic linear models," *Journal of Time Series Analysis*, vol. 15, no. 2, pp. 183–202, 1994.

[180] P. DE JONG and N. SHEPHARD, "The simulation smoother for time series models," *Biometrika*, vol. 82, no. 2, pp. 339–350, 1995.

[181] W. Fong, S. Godsill, A. Doucet, and M. West, "Monte carlo smoothing with application to audio signal enhancement," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 438–449, 2002.

[182] F. Lindsten and T. Schön, "Rao-Blackwellized particle smoothers for mixed linear/nonlinear state-space models," Technical Report LiTH-ISY-R-2018, Linköpings Universitet, 2011.

[183] S. Särkkä, P. Bunch, and S. Godsill, "A backward-simulation based Rao-Blackwellized particle smoother for conditionally linear Gaussian models," in *Proceedings of the 16th IFAC Symposium on System Identification, Brussels, Belgium*, 2012.

[184] P. Bunch and S. Godsill, "Particle smoothing algorithms for variable rate models," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1663–1675, 2013.

[185] B. Malkiel, "The efficient market hypothesis and its critics," *The Journal of Economic Perspectives*, vol. 17, no. 1, pp. 59–82, 2003.

[186] E. Fama, "Efficient capital markets: II," *Journal of Finance*, vol. 46, no. 5, pp. 1575–1617, 1991.

[187] N. Jegadeesh and S. Titman, "Returns to buying winners and selling losers: Implications for stock market efficiency," *Journal of Finance*, vol. 48, no. 1, pp. 65–91, 1993.

[188] D. Vayanos and P. Woolley, "An institutional theory of momentum and reversal," *Working Paper, National Bureau of Economic Research*, 2008.

[189] D. Lesmond, M. Schill, and C. Zhou, "The illusory nature of momentum profits," *Journal of Financial Economics*, vol. 71, no. 2, pp. 349–380, 2004.

[190] E. Fama, "Efficient capital markets: A review of theory and empirical work," *The Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[191] L. Menkhoff and M. P. Taylor, "The obstinate passion of foreign exchange professionals: Technical analysis," *Journal of Economic Literature*, vol. 45, no. 4, pp. pp. 936–972, 2007.

[192] A. Lo and A. MacKinlay, *A Non-Random Walk Down Wall Street*. Princeton University Press, 2001.

[193] A. Lo, H. Mamaysky, and J. Wang, "Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation," *Journal of Finance*, vol. 55, no. 4, pp. 1705–1765, 2000.

[194] R. Levich and L. Thomas, "The significance of technical trading-rule profits in the foreign exchange market: a bootstrap approach," *Journal of International Money and Finance*, vol. 12, no. 5, pp. 451–474, 1993.

[195] H. Hong, T. Lim, and J. Stein, "Bad news travels slowly: Size, analyst coverage, and the profitability of momentum strategies," *Journal of Finance*, vol. 55, no. 1, pp. 265–295, 2000.

[196] N. Jegadeesh and S. Titman, "Profitability of momentum strategies: An evaluation of alternative explanations," *The Journal of Finance*, vol. 56, no. 2, pp. 699–720, 2001.

[197] K. Chan, A. Hameed, and W. Tong, "Profitability of momentum strategies in the international equity markets," *Journal of Financial and Quantitative Analysis*, vol. 35, no. 2, pp. 153–172, 2000.

[198] J. Okunev and D. White, "Do momentum-based strategies still work in foreign currency markets?," *Journal of Financial and Quantitative Analysis*, vol. 38, no. 2, pp. 425–447, 2003.

[199] W. Fung and D. Hsieh, "The risk in hedge fund strategies: Theory and evidence from trend followers," *Review of Financial Studies*, vol. 14, pp. 313–341, 2001.

[200] J. Miffre and G. Rallis, "Momentum strategies in commodity futures markets," *Journal of Banking & Finance*, vol. 31, pp. 1863–1886, 2007.

[201] R. Shiller, *Irrational exuberance*. Princeton University Press, 2005.

[202] T. Johnson, "Rational momentum effects," *Journal of Finance*, vol. 57, no. 2, pp. 585–608, 2002.

[203] S. Schulmeister, "Profitability of technical stock trading: Has it moved from daily to intraday data?," *Review of Financial Economics*, vol. 18, no. 4, pp. 190–201, 2009.

[204] X. Rong Li and V. Jilkov, "Survey of maneuvering target tracking, part I: Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333 – 1364, 2003.

[205] E. Fama, "The behavior of stock-market prices," *Journal of Business*, vol. 38, no. 1, 1965.

[206] D. M. Guillaume, M. M. Dacorogna, R. R. Dave, U. A. Muller, R. B. Olsen, and O. V. Pictet, "From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets," *Finance and Stochastics*, vol. 1, pp. 95–129, 1997.

[207] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.

[208] R. F. Engle, "The econometrics of ultra-high-frequency data," *Econometrica*, vol. 68(1), pp. 1–22, 2000.

[209] P. A. Mykland and L. Zhang, *The Econometrics of High Frequency Data (in Statistical methods for stochastic differential equations)*, ch. 2, pp. 109–190. Chapman & Hall, 2009.

[210] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *Journal of Econometrics*, vol. 31, no. 3, pp. 307–327, 1986.

[211] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973.

[212] S. Heston, "A closed-form solution for options with stochastic volatility with applications to bond and currency options," *Review of Financial Studies*, vol. 6, no. 2, pp. 327–343, 1993.

[213] S. Kou, "A jump-diffusion model for option pricing," *Management Science*, vol. 48, pp. 1086–1101, 2002.

[214] O. E. Barndorff-Nielsen and N. Shephard, *Modelling by Levy processes for financial econometrics (in Levy processes: Theory and applications)*, ch. 5.1, pp. 283–318. Springer/Birkhauser, 2001.

[215] L. Sun and C. Stivers, "Cross-sectional return dispersion and time variation in value and momentum premium," *Journal of Financial and Quantitative Analysis*, vol. 45, pp. 987–1014, 2010.

[216] M. Sorenson, *Statistical Inference in Stochastic Processes*, ch. 3, pp. 67–105. Marcel Dekker, 1991.

[217] C. Ramezani and Y. Zeng, "Maximum likelihood estimation of asymmetric jump-diffusion processes: Application to security prices," *Working paper, Department of Statistics, University of Wisconsin, Maddison (available at SSRN 606361)*, 1998.

[218] C. A. Ramezani and Y. Zeng, "Maximum likelihood estimation of the double exponential jump-diffusion process," *Annals of Finance*, vol. 3, no. 4, pp. 487–507, 2007.

[219] S. Beckers, "A note on estimating the parameters of the diffusion-jump model of stock returns," *Journal of Financial and Quantitative Analysis*, vol. 16, no. 1, pp. 127–140, 1981.

[220] F. B. Hanson and J. Westman, "Jump-diffusion stock return models in finance: Stochastic process density with uniform-jump amplitude," in *Proceedings of 15th International Symposium on Mathematical Theory of Networks and Systems*, vol. 7, 2002.

[221] J. S. Liu, "The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem," *Journal of the American Statistical Association*, vol. 89, no. 427, pp. 958–966, 1994.

[222] R. Waagepetersen and D. Sorensen, "A tutorial on reversible jump mcmc with a view toward applications in qtl-mapping," *International Statistical Review*, vol. 69, no. 1, pp. 49–61, 2001.

[223] S. Godsill and T. Clapp, *Improvement strategies for Monte Carlo particle filters*, pp. 139–158. Springer, 2001.

[224] N. Whiteley, "Response to "Particle Markov chain Monte Carlo methods" by C. Andrieu, A. Doucet and R. Holenstein," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[225] S. Godsill, "Response to "Particle Markov chain Monte Carlo methods" by C. Andrieu, A. Doucet and R. Holenstein," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[226] N. Whiteley, A. M. Johansen, and S. Godsill, "Efficient Monte Carlo filtering for discretely observed jumping processes," in *IEEE/SP 14th Workshop on Statistical Signal Processing, 2007. SSP'07*, pp. 89–93, IEEE, 2007.

[227] N. Whiteley, A. M. Johansen, and S. Godsill, "Monte Carlo filtering of piecewise deterministic processes," *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, 2011.

[228] N. Chopin and S. S. Singh, "On the particle Gibbs sampler," *arXiv preprint arXiv:1304.1887*, 2013.

[229] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[230] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *International Joint Conference on Artificial Intelligence*, vol. 18, pp. 1151–1156, 2003.

[231] T. Kirubarajan, Y. Bar-Shalom, K. Pattipati, and I. Kadar, "Ground target tracking with variable structure IMM estimator," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 26–46, 2000.

[232] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[233] C. S. Agate and K. J. Sullivan, "Road-constrained target tracking and identification using a particle filter," in *Proceedings of SPIE*, vol. 5204, p. 532, 2003.

[234] D. Salmond, M. Clark, R. Vinter, and S. Godsill, "Ground target modelling, tracking and prediction with road networks," in *Proceedings of 10th International Conference on Information Fusion, 2007*, pp. 1–8, IEEE, 2007.

[235] F. Gustafsson, U. Orguner, T. B. Schön, P. Skoglar, and R. Karlsson, *Navigation and tracking of road-bound vehicles using map support (in Handbook of intelligent vehicles)*, ch. 16, pp. 397–434. Springer, 2012.

[236] S. S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[237] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[238] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1778–1784, IEEE, 1988.

[239] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 5, pp. 615–620, 2000.

[240] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, pp. 256–263, IEEE, 2000.

[241] E. Masehian and D. Sedighizadeh, "Classic and heuristic approaches in robot motion planning-a chronological review," *World Academy of Science, Engineering and Technology*, vol. 23, pp. 101–106, 2007.

[242] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 99, p. 80, 2006.

[243] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 108–117, 2006.

[244] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[245] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, AAAI, 2002.

[246] K. Murphy, "Bayesian map learning in dynamic environments," *Advances in Neural Information Processing Systems (NIPS)*, vol. 12, pp. 1015–1021, 1999.

[247] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for

grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[248] A. Eliazar and R. Parr, "DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *International Joint Conference on Artificial Intelligence*, vol. 18, pp. 1135–1142, 2003.

[249] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3d modeling of indoor environments," *International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[250] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[251] A. Ranganathan, E. Menegatti, and F. Dellaert, "Bayesian inference in the space of topological maps," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 92–107, 2006.

[252] L. Liao, D. J. Patterson, D. Fox, and H. Kautz, "Learning and inferring transportation routines," *Artificial Intelligence*, vol. 171, no. 5, pp. 311–331, 2007.

[253] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse gps traces for map inference: comparison of approaches," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 669–677, ACM, 2012.

[254] E. Solak, R. Murray-Smith, W. Leithead, D. Leith, and C. Rasmussen, "Derivative observations in Gaussian process models of dynamic systems," in *Advances in Neural Information Processing Systems*, MIT Press, 2003.

[255] A. Doucet, N. De Freitas, and N. Gordon, eds., *Sequential Monte Carlo methods in practice*. Springer, 2001.

[256] V. V. Williams, "Multiplying matrices faster than Coppersmith-Winograd," in *Proceedings of the 44th symposium on Theory of Computing*, pp. 887–898, ACM, 2012.

[257] L. Csató, M. Opper, and O. Winther, "TAP Gibbs free energy, belief propagation and sparsity," *Advances in Neural Information Processing Systems*, vol. 14, pp. 657–663, 2001.

[258] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[259] N. D. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," *Advances in Neural Information Processing Systems*, vol. 15, no. 15, pp. 609–616, 2002.

[260] M. Seeger, C. K. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *Proceedings of Workshop on AI and Statistics*, vol. 9, 2003.

[261] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, 2006.

[262] J. Moreno and H. Jennings, "Statistics of social configurations," *Sociometry*, vol. 1, pp. 342–374, 1938.

[263] G. Iori, G. De Masi, O. Precup, G. Gabbi, and G. Caldarelli, "A network analysis of the Italian overnight money market," *Journal of Economic Dynamics and Control*, vol. 32, no. 1, pp. 259–278, 2008.

[264] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[265] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[266] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. Di Bernardo, "How to infer gene networks from expression profiles," *Molecular Systems Biology*, vol. 3, no. 1, p. 78, 2007.

[267] B. Ellis and W. Wong, "Learning causal Bayesian network structures from experimental data," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 778–789, 2008.

[268] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-target tracking-linking identities using Bayesian network inference," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2187–2194, IEEE Computer Society, 2006.

[269] A. Gning, L. Mihaylova, S. Maskell, S. Pang, and S. Godsill, "Group object structure and state estimation with evolving networks and Monte Carlo methods," *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1383–1396, 2011.

[270] S. Pang, J. Li, and S. Godsill, "Detection and tracking of coordinated groups," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 1, pp. 472–502, 2011.

[271] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics*, vol. 18, no. 2, pp. 261–274, 2002.

[272] T. Chen, H. L. He, and G. M. Church, "Modeling gene expression with differential equations," in *Pacific Symposium on Biocomputing*, vol. 4, p. 4, 1999.

[273] H. De Jong, "Modeling and simulation of genetic regulatory systems:

a literature review," *Journal of Computational Biology*, vol. 9, no. 1, pp. 67–103, 2002.

[274] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[275] N. Friedman, "The Bayesian structural EM algorithm," in *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pp. 129–138, 1998.

[276] D. Heckerman, D. Geiger, and D. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[277] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[278] P. Giudici and R. Castelo, "Improving Markov chain Monte Carlo model search for data mining," *Machine Learning*, vol. 50, no. 1-2, pp. 127–158, 2003.

[279] D. Madigan, J. York, and D. Allard, "Bayesian graphical models for discrete data," *International Statistical Review/Revue Internationale de Statistique*, vol. 63, no. 2, pp. 215–232, 1995.

[280] N. Friedman and D. Koller, "Being Bayesian about network structure. a Bayesian approach to structure discovery in Bayesian networks," *Machine Learning*, vol. 50, no. 1-2, pp. 95–125, 2003.

[281] K. P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, 2002.

[282] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1991.

[283] S. J. Godsill and P. J. W. Rayner, *Digital Audio Restoration: A Statistical Model-Based Approach*. Berlin: Springer, ISBN 3 540 76222 1, Sept. 1998.

[284] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 139–147, 1998.

[285] S. Y. Kim, S. Imoto, and S. Miyano, "Inferring gene networks from time series microarray data using dynamic Bayesian networks," *Briefings in Bioinformatics*, vol. 4, no. 3, pp. 228–235, 2003.

[286] S. Kim, S. Imoto, and S. Miyano, "Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data," *Biosystems*, vol. 75, no. 1-3, pp. 57–65, 2004.

[287] S. Imoto, S. Kim, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, and S. Miyano, "Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network," *Journal of Bioinformatics and Computational Biology*, vol. 1, no. 2, pp. 231–252, 2003.

[288] N. Sugimoto and H. Iba, "Inference of gene regulatory networks by means of dynamic differential Bayesian networks and nonparametric regression," *Genome Informatics Series*, vol. 15, no. 2, p. 121, 2004.

[289] E. R. Morrissey, M. A. Juárez, K. J. Denby, and N. J. Burroughs, "Inferring the time-invariant topology of a nonlinear sparse gene regulatory network using fully Bayesian spline autoregression," *Biostatistics*, vol. 12, no. 4, pp. 682–694, 2011.

[290] D. Clark and S. Godsill, "Group target tracking with the Gaussian mixture probability hypothesis density filter," in *Proceedings of 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*, pp. 149 –154, 2007.

[291] J. Koch, "Bayesian approach to extended object and cluster tracking using random matrices," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 3, pp. 1042–1059, 2008.

[292] F. Septier, S. K. Pang, S. Godsill, and A. Carmi, "Tracking of coordinated groups using marginalised MCMC-based particle algorithm," in *Proceedings of IEEE Aerospace Conference*, pp. 1–11, IEEE, 2009.

[293] S. Pang, S. Godsill, J. Li, and F. Septier, *Sequential inference for dynamically evolving groups of objects (in Inference and learning in dynamic models)*. Cambridge University Press, 2011.

[294] A. Y. Carmi, L. Mihaylova, A. Gning, P. Gurfil, and S. J. Godsill, "Monte Carlo-based Bayesian group object tracking and causal reasoning," in *Advances in Intelligent Signal Processing and Data Mining*, pp. 7–53, Springer, 2013.

[295] G. Golub and C. Van Loan, *Matrix computations*. Johns Hopkins University Press, 1996.

[296] F. Hoti, "Kernel regression via binned data," Tech. Rep. Research Reports C38, Rolf Nevanlinna Institute, Helsinki, Finland, February 2001.

[297] J. Fan and J. S. Marron, "Fast implementations of nonparametric curve estimators," *Journal of Computational and Graphical Statistics*, vol. 3, no. 1, pp. 35–56, 1994.

[298] P. Hall and M. Ward, "On the accuracy of binned kernel density estimators," *Journal of Multivariate Analysis*, vol. 56, pp. 165–184, 1996.

[299] M. Jones, "A variation on local linear regression," *Statistica Sinica*, vol. 7, pp. 1171–1180, 1997.

[300] F. Hoti and L. Holmström, "On the estimation error in binned local linear regression," *Journal of Nonparametric Statistics*, vol. 15, no. 4-5, pp. 625–642, 2003.

[301] S. Kou, Q. Zhou, and W. H. Wong, "Equi-energy sampler with applications in statistical inference and statistical mechanics," *The Annals of Statistics*, vol. 34, no. 4, pp. 1581–1619, 2006.

[302] N. SHEPHARD and M. K. PITT, "Likelihood analysis of non-Gaussian measurement time series," *Biometrika*, vol. 84, no. 3, pp. 653–667, 1997.

[303] P. J. Wolfe, S. J. Godsill, and W. Ng, "Bayesian variable selection and regularisation for time-frequency surface estimation," *Journal of the Royal Statistical Society, Series B*, vol. 66, no. 3, pp. 575–589, 2004. Read paper (with discussion).

[304] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 27, no. 2, pp. 113–120, 1979.

[305] J. Erkelens and R. Heusdens, "Tracking of nonstationary noise based on data-driven recursive noise power estimation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1112–1123, 2008.

[306] S. Godsill, "The shifted inverse-gamma model for noise floor estimation in archived audio recordings," *Applied Signal Processing*, 2010. Special Issue on Preservation of Ethological Recordings.

[307] I. Soon, S. Koh, and C. Yeo, "Noisy speech enhancement using discrete cosine transform," *Speech Communication*, vol. 24, no. 3, pp. 249–257, 1998.

[308] P. Wolfe and S. J. Godsill, "Interpolation of missing data values for audio signal restoration using a Gabor regression model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2005.

[309] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[310] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[311] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[312] H. Feichtinger and T. Strohmer, *Gabor analysis and algorithms: Theory and applications*. Birkhauser, 1998.

[313] R. Balian, "Un principe d'incertitude fort en théorie du signal ou en mécanique quantique," *Comptes Rendus de l'Académie des Sciences, Paris*, vol. 292, no. 2, pp. 1357–1361, 1981.

[314] F. Low, "Complete sets of wave packets," in *A Passion for Physics–Essays in Honor of Geoofrey Chew*, pp. 17–22, World Scientific, 1985.

[315] S. Qian and D. Chen, "Discrete Gabor transform," *IEEE Transactions on Signal Processing*, vol. 41, no. 7, pp. 2429–2438, 1993.

[316] D. F. Andrews and C. L. Mallows, "Scale mixtures of normal distributions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 1, pp. 99–102, 1974.

[317] S. Choy and A. Smith, "Hierarchical models with scale mixtures of normal distributions," *Test*, vol. 6, no. 1, pp. 205–221, 1997.

[318] D. L. Ermak and J. A. McCammon, "Brownian dynamics with hydrodynamic interactions," *The Journal of Chemical Physics*, vol. 69, p. 1352, 1978.

[319] D. J. Higham, "An algorithmic introduction to numerical simulation of stochastic differential equations," *SIAM Review*, vol. 43, no. 3, pp. 525–546, 2001.

[320] W. Van Gunsteren and H. Berendsen, "Algorithms for Brownian dynamics," *Molecular Physics*, vol. 45, no. 3, pp. 637–647, 1982.

[321] A. Iniesta and J. G. de la Torre, "A second-order algorithm for the simulation of the Brownian dynamics of macromolecular models," *The Journal of Chemical Physics*, vol. 92, p. 2015, 1990.

[322] A. Brańka and D. Heyes, "Algorithms for Brownian dynamics simulation," *Physical Review E*, vol. 58, no. 2, p. 2611, 1998.

[323] E. Hershkovitz, "A fourth-order numerical integrator for stochastic Langevin equations," *The Journal of Chemical Physics*, vol. 108, p. 9253, 1998.

[324] H. A. Forbert and S. A. Chin, "Fourth-order algorithms for solving the multivariable Langevin equation and the Kramers equation," *Physical Review E*, vol. 63, no. 1, p. 016703, 2000.

[325] K. Burrage, I. Lenane, and G. Lythe, "Numerical methods for second-order stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 29, no. 1, pp. 245–264, 2007.

[326] O. Elerian, S. Chib, and N. Shephard, "Likelihood inference for discretely observed nonlinear diffusions," *Econometrica*, vol. 69, no. 4, pp. 959–993, 2001.

[327] B. Eraker, "MCMC analysis of diffusion models with application to finance," *Journal of Business and Economic Statistics*, vol. 19, no. 2, pp. 177–191, 2001.

Lauda finem.