Sequential Monte Carlo Methods for High-Dimensional State Space Models

Submitted for the Degree of Doctor of Philosophy in Statistical Science

UCL

Neil Foster

Declaration

I, Neil Foster, confirm that the work presented in my thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

This thesis focuses on developing methodologies to approximate the filtering distribution in partially observed state space models, with particular attention to addressing particle degeneracy issues that arise in high-dimensional state spaces and when dealing with informative observations.

Chapter 1 introduces the challenge of sequential inference within state space models, along with an overview of the Sequential Monte Carlo (SMC) methodology. The chapter also presents the bootstrap filter as a tool for approximating the filtering distribution, discussing its limitations and providing background information.

Chapter 2 reviews three popular existing methodologies for filtering discretely observed stochastic differential equations (SDEs).

Chapters 3 and 4 propose a new methodology for filtering partially observed SDEs. This approach combines likelihood-informed proposals, tempering steps targeting intermediate distributions between prior and posterior densities, and mutations through Markov Chain Monte Carlo (MCMC) steps to enhance particle diversity. The methodology's effectiveness is evaluated through a small-scale experiment involving a double-well potential process.

Chapter 5 applies filtering techniques to a tsunami wave field model, treated as a discrete-time state space model. The optimal filter can be derived under a linear observation scheme with Gaussian noise processes for both observations and process noise. We present an efficient approach for com-

puting the optimal filter using the Fast Fourier Transform, which enables efficient sampling of conditional Gaussian random fields and rapid matrix-vector multiplication. Numerical experiments demonstrate the accuracy and efficiency of the proposed method.

Chapter 6 extends the methodology from Chapter 3 to the filtering of a tsunami wave field model represented as a stochastic partial differential equation. This complex setting is tested through experiments that examine the algorithm's robustness with respect to both spatial resolution and the informativeness of observations.

Impact statement

The work presented in this thesis makes contributions to the field of sequential Monte Carlo methods, specifically focusing on high-dimensional state space models with highly informative observations.

The potential applications of this research exist in domains where data assimilation in complex systems is necessary. One example covered in the thesis is the application of the particle filtering method to tsunami wavefield data assimilation. Here the method demonstrates its potential for predicting and monitoring the evolution of tsunami wavefields, which is important for early warning systems, risk mitigation strategies, and public safety. By improving the robustness and computational efficiency of data assimilation techniques in this context, the developed method could inform public policy and enhance the capability of authorities to deliver timely, data-driven responses to natural disasters, potentially saving lives and reducing economic losses.

Acknowledgements

I am deeply grateful to my supervisor, Alexandros Beskos, for his exceptional patience and insightful guidance.

To my family, friends, and partner, thank you for your constant support and belief in me throughout challenging times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Stochastic processes are widely used to model complex real-world phenomena. Their applications span various fields, including biology [Wilkinson, 2006, Goel and Richter-Dyn, 2016], physics [Van Kampen, 1992, Coffey and Kalmykov, 2012, Sobczyk, 2013], neuroscience [Tuckwell, 1989], engineering [Wong and Hajek, 2012, Sobczyk, 2013], and financial market analysis [Rolski et al., 2009, Kijima, 2002, Marsh and Rosenfeld, 1983].

In many of these systems, data arrives sequentially, and we aim to make inferences each time a new set of observations is received. Additionally, this data often contains observation errors that must be accounted for. While traditional Markov Chain Monte Carlo (MCMC) methods can be used to make these inferences, they require sampling from the initial time to the current time each time we receive new data and take into account all previous data points at each time instance, this is impractical for online inference. To overcome these limitations, sequential Monte Carlo (SMC) methods have been developed.

SMC methods allow us to retain information about the system and update our posterior as data arrives, avoiding some of the impracticalities associated with MCMC methods and significantly reducing computational cost. Additionally, when compared to other online estimation algorithms, such as the Kalman filter, we require fewer assumptions about both the underlying dynamical system and error function to make estimates about the current state of the system. These methods can be applied to a wider variety of problems.

SMC methods do have their own limitations, specifically in the small noise and high dimensional settings. We aim to improve on some of these methods, with a focus on techniques applied in the setting of a continuous time stochastic system observed at discrete time periods.

1.1 Hidden Markov models

Hidden Markov models (HMM) are used to model the probability distribution of the state of some system where the state of the system is hidden from the observer and the system satisfies the Markov property; that is, given the state of the system at time t, the state of the system at time $t+1$ depends only on the state of the system at time t and is independent of all states prior to that.

A HMM is defined by a latent, underlying Markov process $\{X_t\}_{t\geqslant 0}$, on some space $\mathcal{X}$ with initial distribution $X_0 \sim f(x_0)$, transition probability density

$$X_t|(X_{t-1} = x_{t-1}) \sim f(x_t|x_{t-1}), \tag{1.1}$$

and $\mathcal{Y}$-valued observations with density

$$Y_t|(X_t = x_t) \sim g(y_t|x_t). \tag{1.2}$$

A simple example of such a system can be seen in Figure 1.1 and a graph showing the dependencies of the system can be seen in Figure 1.2.

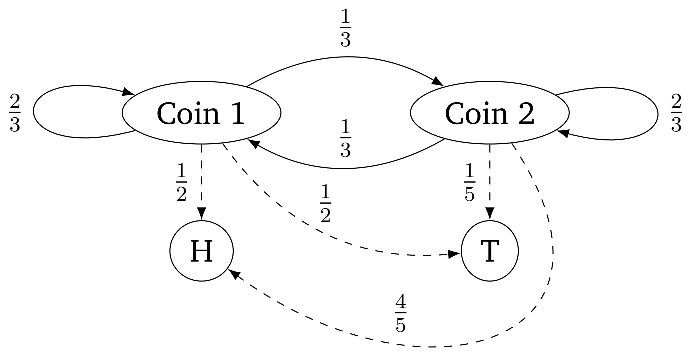


Figure 1.1: An illustration of the probabilities underlying a simple HMM. There are two coins, coin 1 is a fair coin and coin 2 is biased. At time 0, a coin is chosen uniformly at random and flipped. At each subsequent time a fair die is rolled, if the result of the roll is greater than 4, the coin is swapped, otherwise the same coin is flipped. In the HMM model an observer would see only the result of the coin toss at each time, the state of the system (in this case, which coin has been flipped) is hidden.

$$\begin{tikzcd}
X_0 \arrow[r] \arrow[d] & \cdots \arrow[r] & X_{t-1} \arrow[r] \arrow[d] & X_t \arrow[r] \arrow[d] & X_{t+1} \arrow[r] \arrow[d] & \cdots \\
Y_0 & & Y_{t-1} & Y_t & Y_{t+1} &
\end{tikzcd}$$

Figure 1.2: A diagram showing the dependence relations in a HMM with underlying Markov process $\{X_t\}_{n \geqslant 0}$ and observations $\{Y_t\}_{n \geqslant 0}$.

1.1.1 The filtering problem

Assume we have a latent, discrete time Markov process, X_t, taking values in $\mathcal{X}$, with transition probabilities $f\left(x_t|x_{t-1}\right)$ and noisy observations, $Y_t \sim g\left(y_t|x_t\right)$. We are interested in recursively estimating the distribution of the state of our latent Markov process at the current time, given the observed information up to and including the current time, i.e. estimating the posterior distributions $\{p\left(x_{0:t}|y_{0:t}\right)\}_{t\geqslant 0}$ (known as the smoothing distributions), and the marginals $\{p\left(x_t|y_{0:t}\right)\}_{t\geqslant 0}$ (known as the filtering distributions), where $x_{i:j} := \{x_i, \ldots, x_j\}, i \leq j$.

We proceed using a Bayesian approach to inference. Our model provides a prior distribution of the latent process,

$$p(x_{0:t}) = f(x_0)\prod_{i=1}^{t} f(x_i|x_{i-1}), \tag{1.3}$$

and likelihood of the observed data

$$p(y_{0:t}|x_{0:t}) = \prod_{i=0}^{t} g(y_i|x_i). \tag{1.4}$$

The smoothing distribution over the hidden process up to time t is given by

$$\begin{aligned} p\left(x_{0:t}|y_{0:t}\right) &= \frac{p(y_{0:t}|x_{0:t})p(x_{0:t})}{p(y_{0:t})} \\ &= \frac{p(y_{0:t}|x_{0:t})p(x_{0:t})}{\int p(y_{0:t}|x_{0:t})p(x_{0:t})dx_{0:t}}, \end{aligned} \tag{1.5}$$

It is simple to show that this distribution (1.5) satisfies the recursion

$$p(x_{0:t}|y_{0:t}) = p(x_{0:t-1}|y_{0:t-1})\frac{f(x_t|x_{t-1})g(y_t|x_t)}{p(y_t|y_{0:t-1})}. \tag{1.6}$$

where the conditional likelihood of the data point at the current time given the data so far is

$$p(y_t|y_{0:t-1}) = \int p(x_{t-1}|y_{0:t-1})g(y_t|x_t)f(x_t|x_{t-1})dx_{t-1:t}, \tag{1.7}$$

and the likelihood of all the data points up to the current time can be calculated recursively as

$$p(y_{0:t}) = p(y_0) \prod_{i=1}^{t} p(y_i|y_{0:i-1}). \tag{1.8}$$

Integrating out $x_{0:t-1}$ from (1.6) gives us a recursion satisfied by the filtering distribution

$$p(x_t|y_{0:t}) = \frac{p(x_t|y_{0:t-1})g(y_t|x_t)}{p(y_t|y_{0:t-1})}, \tag{1.9}$$

where

$$p(x_t|y_{0:t-1}) = \int p(x_{t-1}|y_{1:t-1})f(x_t|x_{t-1})dx_{t-1}. \tag{1.10}$$

The recursions (1.9) and (1.10) are commonly referred to as the “update” step and “prediction” step respectively. While these recursions look simple, in the majority of cases we cannot evaluate the integrals involved. We could use Markov chain Monte Carlo (MCMC) or regular importance sampling methods to estimate the posterior, but we would like to be able to update the distributions sequentially as new data arrives. In both MCMC and regular importance sampling, old samples cannot be reused when we get a new data point and we must run brand new simulations to obtain a new estimate for the posterior. Neither method is well suited for online estimation in problems of this nature, so particle filtering methods were developed to provide approximations to the distributions of interest. These methods are part of a more general class of methods known as Sequential Monte Carlo (SMC) methods.

1.2 Sequential Monte Carlo Methods

SMC methods are a subset of Monte Carlo methods, designed to sample from sequence of target densities. We begin this section with a brief refresher on the principles of Monte Carlo sampling and Importance Sampling (IS), going on to introduce Sequential Importance Sampling (SIS) and the generic SMC algorithm.

1.2.1 Basic Monte Carlo Sampling

Suppose we wanted to approximate some target density, $p_t(x_{0:t})$, defined on the product space $\mathcal{X}^{t+1}$. If we are able to sample N independent and identically distributed (i.i.d.) random variables $\{X^i_{0:t}\}_{i=1}^N$ according to $p_t(x_{0:t})$ then an empirical estimate of the distribution is given by

$$\widehat{p}_t(x_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta_{X^i_{0:t}}(x_{0:t}), \tag{1.11}$$

where $\delta_{X^i_{0:t}}(x_{0:t})$ denotes the Dirac delta mass located at $X^i_{0:t}$. More precisely, the expectation of any function, $\varphi : \mathcal{X}^{t+1} \to \mathbb{R}$, integrable with respect to $p_t(x_{0:t})$,

$$\mathbb{E}_{p_t}(\varphi(X_{0:t})) := \int \varphi(x_{0:t}) p_t(x_{0:t}) dx_{0:t}, \tag{1.12}$$

is simply approximated by

$$\int \varphi(x_{0:t}) \widehat{p}_t(x_{0:t}) dx_{0:t} = \frac{1}{N}\sum_{i=1}^{N} \varphi(X^i_{0:t}). \tag{1.13}$$

and the strong law of large numbers says

$$\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{0:t}^{i}) \xrightarrow[N\to\infty]{\text{a.s.}} \mathbb{E}_{p_t}(\varphi(X_{0:t})), \tag{1.14}$$

where $\xrightarrow{\text{a.s.}}$ denotes convergence in an almost surely sense.

It is clear to see the estimator is unbiased

$$\mathbb{E}_{p_t}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{0:t}^{i})\right) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{p_t}\left(\varphi(X_{0:t}^{i})\right) = \mathbb{E}_{p_t}(\varphi(X_{0:t})), \tag{1.15}$$

and if $\mathrm{Var}(\varphi(X_{0:t}))$ is finite, the variance of the estimator is given by

$$\mathrm{Var}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{0:t}^{i})\right) = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}(\varphi(X_{0:t})) = \frac{1}{N}\mathrm{Var}(\varphi(X_{0:t})). \tag{1.16}$$

Furthermore, we can obtain a central limit theorem

$$\sqrt{N}\left(\frac{1}{N}\sum_{i=1}^{N}\varphi(X_{0:t}^{i}) - \mathbb{E}_{p_t}(\varphi(X_{0:t}))\right) \xrightarrow[N\to\infty]{\text{d}} \mathcal{N}\left(0, \mathrm{Var}(\varphi(X_{0:t}))\right), \tag{1.17}$$

where $\xrightarrow{\text{d}}$ denotes convergence in distribution.

Following the standard Monte Carlo approach, we can construct (1.13)—an unbiased estimate of (1.12)—using a set of independent and identically distributed (i.i.d.) samples drawn from the distribution $p_t(x_{0:t})$. The standard error of this estimate is independent of the dimensionality of the space, $\mathcal{X}^{t+1}$, and decreases proportionally to the square root of the number of samples. However, in many cases, efficient sampling from the target distribution $p_t(x_{0:t})$ is not feasible, requiring an alternative approach.

1.2.2 Importance Sampling

Importance Sampling addresses this problem by introducing an importance (or proposal) distribution, $q_t(x_{0:t})$, where the support of $p_t(x_{0:t})$ is a subset of the support of $q_t(x_{0:t})$.

Assume now that we cannot draw samples from $p_t(x_{0:t})$ and that it is only known up to a normalising constant, that is,

$$p_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{Z_t} \tag{1.18}$$

where $\gamma_t : \mathcal{X}^{t+1} \longrightarrow \mathbb{R}^+$ is known pointwise and the normalising constant,

$$Z_t = \int \gamma_t(x_{0:t}) dx_{0:t}, \tag{1.19}$$

is unknown.

Now let $w_t(x_{0:t})$ be the unnormalised importance weight

$$w_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})}. \tag{1.20}$$

From (1.18), (1.19) and (1.20) we get the identities

$$p_t(x_{0:t}) = \frac{w_t(x_{0:t}) q_t(x_{0:t})}{Z_t} \tag{1.21}$$

$$Z_t = \int w_t(x_{0:t}) q_t(x_{0:t}) dx_{0:t}. \tag{1.22}$$

If we choose our proposal distribution such that we can easily draw N i.i.d. samples $\{X_{0:t}^i\}_{i=1}^N$ according to $q_t(x_{0:t})$, we can use the resulting Monte

Carlo approximation of $q_t(x_{0:t})$ with (1.21) and (1.22) to obtain estimates of the target distribution and normalising constant

$$\widehat{p}_t(x_{0:t}) = \sum_{i=1}^{N} \tilde{w}_t^i \delta_{X_{0:t}^i}(x_{0:t}) \tag{1.23}$$

$$\widehat{Z}_t = \frac{1}{N} \sum_{i=1}^{N} w_t(X_{0:t}^i) \tag{1.24}$$

where the normalised importance weights, $\tilde{w}_t^i$, are given by

$$\tilde{w}_t^i = \frac{w_t(X_{0:t}^i)}{\sum_{j=1}^{N} w_t(X_{0:t}^j)}. \tag{1.25}$$

It is clear to see that, $\widehat{Z}_t$, the approximation to the normalising constant is unbiased

$$\begin{aligned} \mathbb{E}_{q_t}\left(\frac{1}{N}\sum_{i=1}^{N} w_t(X_{0:t}^i)\right) &= \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{q_t}\left(\frac{\gamma_t(X_{0:t}^i)}{q_t(X_{0:t}^i)}\right) \\ &= \frac{1}{N}\sum_{i=1}^{N} \int \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} q_t(x_{0:t}) dx_{0:t} \\ &= Z_t. \end{aligned} \tag{1.26}$$

An approximation of the expectation of some p_t-integrable function is given by

$$\mathbb{E}_{p_t}(\varphi) \approx \int \varphi(x_{0:t})\widehat{p}_t(x_{0:t}) dx_{0:t} = \sum_{i=1}^{N} \tilde{w}_t^i \varphi(X_{0:t}^i). \tag{1.27}$$

As this estimate (1.27) is the ratio of two estimates, it is biased for finite N (note that if the normalising constant is known then we can construct an unbiased estimator), but under weak assumptions the strong law of large numbers applies [Geweke, 1989] and we have

$$\sum_{i=1}^{N} \tilde{w}_t^i \varphi(X_{0:t}^i) \xrightarrow[N\to\infty]{\text{a.s.}} \mathbb{E}_{p_t}(\varphi). \tag{1.28}$$

Under further assumptions (see Geweke [1989] for details), we can obtain a central limit theorem

$$\sqrt{N}\left(\sum_{i=1}^{N} \tilde{w}_t^i \varphi(X_{0:t}^i) - \mathbb{E}_{p_t}(\varphi)\right) \xrightarrow[N\to\infty]{\text{d}} \mathcal{N}\left(0, \sigma_t^2\right), \tag{1.29}$$

where

$$\sigma_t^2 = \frac{1}{N}\int \frac{p_t^2(x_{0:t})}{q_t(x_{0:t})}(\varphi(x_{0:t}) - \mathbb{E}_{p_t}(\varphi))^2 dx_{0:t}. \tag{1.30}$$

IS enables the approximation of integrals with respect to distributions that are difficult to sample from directly. However, the computational complexity of sampling from a proposal distribution $q_t(x_{0:t})$ generally increases with t . This growing complexity over time renders IS unsuitable for recursive estimation tasks, such as estimating the filtering distribution of a HMM as data arrives sequentially. For these tasks, a method with computational complexity independent of t at each time step is required.

1.2.3 Sequential Importance Sampling

Sequential Importance Sampling (SIS) is a restricted form of Importance Sampling (IS), characterized by the requirement to choose an importance distribution of the form:

$$q_t(x_{0:t}) = q_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1}). \tag{1.31}$$

Iterating (1.31) over t, we obtain:

$$q_t(x_{0:t}) = q_0(x_0) \prod_{k=1}^{t} q_t(x_t|x_{0:t-1}). \tag{1.32}$$

So to generate a sample, $X^i_{0:t} \sim q_t(x_{0:t})$, we sample $X^i_0 \sim q_0(x_0)$ at time 0 and then recursively sample $X^i_k \sim q_k(x_k|X^i_{0:k-1})$ for each time, k, up to time t.

An importance distribution of the form (1.31) is necessary as it allows us to compute the unnormalised importance weights recursively,

$$\begin{aligned} w_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} \\ &= \frac{\gamma_{t-1}(x_{0:t-1})}{q_{t-1}(x_{0:t-1})} \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1})} \\ &= w_{t-1}(x_{0:t-1}) \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1})}. \end{aligned} \tag{1.33}$$

The recursive updating of the importance weights is the defining feature of SIS, enabling the propagation of samples from time $t-1$ to time t and allowing for the recursive estimation of the posterior $p_t(x_{0:t})$. The SIS algorithm is outlined in Algorithm 1.1.

SIS is a special case of the more general IS, so empirical estimates $\widehat{p}_t(x_{0:t})$ and $\widehat{Z}_t$ are as in IS, (1.23) and (1.24) respectively. The estimates of the normalisation constant and expectation of a test function inherit the bias and convergence properties outlined for those estimates based on IS.

SIS is an appealing method and in most cases can be implemented with a computational cost that scales linearly in the number of particles and

Algorithm 1.1: Sequential importance sampling

```
Initialise particles:
  1. For $i = 1, \dots, N$:
       (i)  Sample $x_0^i \sim q_0(x_0)$.
       (ii) Compute unnormalised importance weights $w_0(x_0^i) = \frac{\gamma_0(x_0^i)}{q_0(x_0^i)}$
            and normalised importance weights $\tilde{w}_0^i \propto w_0(x_0^i)$.
  2. Set $t = 1$.

Importance sampling:
  3. For $i = 1, \dots, N$
       (i)  Sample $x_t^i \sim q(x_t | x_{0:t-1}^i)$.
       (ii) Compute unnormalised importance weights,

            $$w_t(x_{0:t}^i) = w_{t-1}(x_{0:t-1}^i) \frac{\gamma_t(x_{0:t}^i)}{\gamma_{t-1}(x_{0:t-1}^i) q_t(x_t^i | x_{0:t-1})}$$

  4. Compute normalised importance weights,

            $$\tilde{w}_t^i = \frac{w_t(x_{0:t}^i)}{\sum_{j=1}^N w_t(x_{0:t}^j)}.$$

  5. Set $t = t + 1$. If $t > T$, proceed. Else, go to (3).

Output:
  6. For $t = 0, \dots, T$:
       (i)  Output $\{x_{0:t}^i, \tilde{w}_t^i\}_{i=1}^N$.
```

number of time steps. However, it is well known that the efficiency of IS scales poorly with the dimension of the space [Gordon et al., 1993, Gilks et al., 1996] - the variance of the estimates increases exponentially in t. As SIS is merely a specific case of IS it will inherit this adverse property.

This problem reveals itself in the distribution of the weights; as t increases the maximum of the weights will approach one while the remaining weights are close to zero. Clearly this situation is inadequate, effectively approximating the target distribution with a single sample. An extra step can be added to the standard SIS algorithm in an attempt to combat this degeneracy.

1.2.4 Resampling

Resampling seeks to multiply the samples with high importance weights and remove the samples whose importance weights are close to zero [Gordon et al., 1993]. Estimates of the posterior from SIS consist of N weighted samples, $\{X_{0:t}^i, \tilde{w}_t^i\}_{i=0}^N$, with an approximation to the posterior given by (1.23). We replace these weighted samples with N unweighted samples, sampled from our IS approximation to the posterior; that is we select N samples with replacement, where the probability of selecting a sample, $X_{0:t}^i$, is equal to its importance weight, $\tilde{w}_t^i$. This gives us an empirical estimate of the estimator $\hat{p}(x_{0:t})$

$$\hat{\hat{p}}(x_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} N_t^i \delta_{X_{0:t}^i}(x_{0:t}), \tag{1.34}$$

where N_t^i is the number of copies of $X_{0:t}^i$ in the resampled set.

We would like to perform this resampling so that we have an unbiased estimator of $\hat{p}(x_{0:t})$. Three common unbiased resampling schemes are described below:

- *Multinomial Resampling* - Draw $u^i \sim U(0,1)$ and find the corresponding k^i such that

$$\sum_{j=1}^{k^i-1} \tilde{w}_t^j \leqslant u^i \leqslant \sum_{j=1}^{k^i} \tilde{w}_t^j \tag{1.35}$$

is satisfied. The set of indices $\{k^i\}_{i=1}^N$ refer to the new samples and our resampled set is $\left\{X_{0:t}^{k^i}\right\}_{i=1}^N$.

- *Stratified Resampling* - Stratify the unit interval into N disjoint intervals, $(0, \frac{1}{N}), \cdots, (1-\frac{1}{N}, 1)$, and draw $u^i \sim U(\frac{i-1}{N}, \frac{i}{N})$. Find corresponding k^i that satisfy (1.35) and proceed as above.

- *Systematic Resampling* - Draw $u^1 \sim U(0, \frac{1}{N})$. For $i > 1$, set

$$u^i = u^1 + \frac{i-1}{N} \tag{1.36}$$

  and again find the corresponding k^i satisfying (1.35) to determine the indices of the resampled set of samples.

Systematic and stratified sampling are often used in practice as they frequently offer improved performance, but convergence can be sensitive to the ordering of the samples [Gerber et al., 2019]. For a detailed discussion of the properties of these resampling schemes see Douc and Cappe [2005].

These resampling schemes multiply the particles with higher weights and eliminate many of the particles with low weights, reducing the sample diversity, a side effect of this is that extra variance is introduced in our estimators. Because of this, we may only want to perform resampling when the weights of the samples have degenerated beyond some threshold. This is often determined by looking at the effective sample size (ESS) [Liu, 2008], defined at time t as

$$\text{ESS}_t = \frac{1}{\sum_{i=1}^N (\tilde{w}_t^i)^2}. \tag{1.37}$$

The ESS can be interpreted as an approximation to the number of independent and identically distributed (IID) samples that we would need from the target distribution to achieve a similar level of estimation accuracy. A common approach is to resample adaptively, only resampling at those time steps where the ESS falls below a pre-determined threshold, often $\frac{N}{2}$.

1.2.5 Sequential Monte Carlo

Sequential Monte Carlo methods can be thought of as SIS with an added resampling step. A repeated sequence of propagating samples forward in time according to a proposal distribution $q_t(x_t|x_{0:t-1})$, followed by (if required) resampling the samples according to their normalised weights before propagating forward to the next time step. A generic SMC algorithm is presented in Algorithm 1.2.

1.3 Particle filtering

We return now to the problem of online estimation of the latent state of a HMM and the filtering problem, as described in section 1.1.1. A common way to approach this problem is by applying the SMC methods introduced in the previous sections, with the target distribution as the state of the latent Markov process dependent on the observed data points, $p_t(x_{0:t}|y_{0:t})$. From this, the marginals can be taken to estimate the filtering distribution, $p_t(x_t|y_{0:t})$.

We are seeking to estimate the posterior of the hidden process as in (1.5), comparing this with (1.18) we can identify the unnormalised distribution $\gamma_t(x_{0:t}) = p(y_{0:t}|x_{0:t})p(x_{0:t})$ and the normalisation constant $Z_t = p(y_{0:t})$. We

Algorithm 1.2: Sequential Monte Carlo (with adaptive resampling)

```
Initialise particles:
  1. For $i = 1, \dots, N$:
       (i) Sample $x_0^i \sim q_0(x_0)$.
       (ii) Compute unnormalised importance weights $w_0(x_0^i) = \frac{\gamma_0(x_0^i)}{q_0(x_0^i)}$
            and normalised importance weights $\tilde{w}_0^i \propto w_0(x_0^i)$.
  2. Set $t = 1$.

Importance sampling:
  3. For $i = 1, \dots, N$
       (i) Sample $x_t^i \sim q_t(x_t | x_{0:t-1}^i)$.
       (ii) Compute unnormalised importance weights,
            $$w_t(x_{0:t}^i) = w_{t-1}(x_{0:t-1}^i) \frac{\gamma_t(x_{0:t}^i)}{\gamma_{t-1}(x_{0:t-1}^i) q_t(x_t^i | x_{0:t-1})}$$
  4. Compute normalised importance weights,
            $$\tilde{w}_t^i = \frac{w_t(x_{0:t}^i)}{\sum_{j=1}^N w_t(x_{0:t}^j)}.$$

Resampling:
  5. Calculate ESS, if resampling condition is met, proceed. Else, go to
     (8).
  6. For $i = 1, \dots, N$:
       (i) Sample $\bar{x}_{0:t}^i$ according to
            $$\widehat{p}_t(x_{0:t}) = \sum_{i=1}^N \tilde{w}_t^i \delta_{x_{0:t}^i}(x_{0:t})$$
  7. Set $\{x_{0:t}^i, \tilde{w}_t^i\}_{i=1}^N = \{\bar{x}_{0:t}^i, \frac{1}{N}\}_{i=1}^N$.
  8. Set $t = t + 1$. If $t > T$, proceed. Else, go to (3).

Output:
  9. For $t = 0, \dots, T$:
       (i) Output $\{x_{0:t}^i, \tilde{w}_t^i\}_{i=1}^N$.
```

choose our proposal distribution to be time homogeneous and dependent only on the state of the Markov process at the previous time step and the data point at the current time, i.e. $q_t(x_t|x_{0:t-1}) = q(x_t|x_{t-1}, y_t)$. Combining this with (1.33) we obtain a recursion for updating the importance weights

$$w_t(x_{0:t}) = w_{t-1}(x_{0:t-1})\frac{g(y_t|x_t)f(x_t|x_{t-1})}{q(x_t|x_{t-1}, y_t)}, \tag{1.38}$$

where $f(x_t|x_{t-1})$ and $g(y_t|x_t)$ are as in the definitions of a HMM, (1.1) and (1.2) respectively.

The application of SMC in the context of filtering is known in the literature as the *particle filter*. A generic particle filtering algorithm, with observations $\{Y_0, Y_1, \cdots, Y_t\}$, is presented in Algorithm 1.3.

1.3.1 Bootstrap particle filter

The bootstrap filter [Gordon et al., 1993] is one of the most widely used filtering algorithms, utilizing the transitional prior $f(x_t|x_{t-1})$ as the proposal distribution. This choice simplifies the update of the importance weights to:

$$w_t \propto w_{t-1}g(y_t|x_t). \tag{1.39}$$

The bootstrap algorithm is applicable to a wide range of models, and its simplicity makes it easy to implement and highly suitable for parallel processing. However, this simplicity comes at the cost of computational inefficiency in more complex applications, particularly as the state space dimension increases or the variance of observation noise decreases. In such cases, a more sophisticated proposal distribution may be required to better

Algorithm 1.3: Particle filter

```
Initialise particles:
  1. For $i = 1, \ldots, N$:
       (i) Sample $x_0^i \sim q(x_0)$.
       (ii) Evaluate unnormalised importance weights $w_0^i = g(y_0|x_0^i)$.
  2. Normalise importance weights and set $t = 1$.

Importance sampling:
  3. For $i = 1, \ldots, N$
       (i) Sample $x_t^i \sim q(x_t|x_{t-1}^i, y_t)$.
       (ii) Evaluate unnormalised importance weights,
            $$w_t^i = w_{t-1}^i \frac{g(y_t|x_t^i) f(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, y_t)}.$$
  4. Normalise importance weights,
            $$\tilde{w}_t^i = \frac{w_t^i}{\sum_{j=1}^N w_t^j}.$$

Resampling:
  5. Calculate ESS.
       (i) If resampling condition is met, proceed. Else, go to (8).
  6. For $i = 1, \ldots, N$:
       (i) Sample $\bar{x}_{0:t}^i$ according to
            $$\widehat{p}(x_{0:t}|y_{0:t}) = \sum_{i=1}^N \delta_{x_{0:t}^i}(x_{0:t}) \tilde{w}_t^i$$
  7. Set $\{x_{0:t}^i, w_t^i\}_{i=1}^N = \{\bar{x}_t^i, \frac{1}{N}\}_{i=1}^N$.
  8. Set $t = t + 1$:
       (i) Set $t = t + 1$. If $t > T$, proceed. Else, go to (3).

Output:
  9. For $t = 0, \ldots, T$:
       (i) Output $\{x_{0:t}^i, w_t^i\}_{i=1}^N$.
```

"guide" the particles toward the target distribution.

1.3.2 Auxiliary particle filter

In the standard particle filter, it's possible to guide particles toward the regions of state space which are more likely given the observed data at the current time by selecting an appropriate proposal, $q(x_t|x_{t-1}, y_t)$, the particles are then resampled according to their weights. This resampling is done without any information about the next observation, y_t. This information could be useful in preserving diversity in the particles by taking into account which particles are likely to produce samples in regions compatible with the next observation.

The *auxiliary particle filter attempts* to mitigate this problem with the introduction of an auxiliary variable at the sampling step of the regular particle filter. This auxiliary variable, $k \in \{1, \cdots, N\}$, corresponds to the index of a particle and is sampled according to some distribution that attributes each particle a weight corresponding to its compatibility with the next observed data point. The new state for a particle is then sampled as the child of the particle corresponding to index k. In attributing particles weight for the index selection we seek to find some approximation $\widehat{p}(y_t|x_{t-1})$, of $p(y_t|x_{t-1})$, where

$$p(y_t|x_{t-1}) = \int g(y_t|x_t) f(x_t|x_{t-1}) dx_t. \tag{1.40}$$

The weight update then becomes

$$w_t^i \propto w_{t-1}^{k_i} \frac{g(y_t|x_t^i) f(x_t^i|x_{t-1}^{k_i})}{\widehat{p}(y_t|x_{t-1}^{k_i}) q(x_t^i|x_{t-1}^{k_i}, y_t)}, \tag{1.41}$$

where k_i is the index sampled for the ith particle, the auxiliary particle filtering algorithm is given in Algorithm 1.4.

It is worth noting that Carpenter et al. [1999] presents a version of the auxiliary particle filter in which the auxiliary weights and standard weights are combined and only a single resampling step takes place per iteration, this approach is generally preferred in the literature. Here we have presented the version of auxiliary particle filtering from Pitt and Shephard [1999], as its structure is closer to that of the random weight particle filter presented in Fearnhead et al. [2008].

1.4 Feynman-Kac models

We introduce Feynman-Kac models Del Moral [2004], which give us an alternative way of thinking about sequential importance sampling. These models consist of two main elements: a Markov chain with its associated probability measure on the state space and sequence of potentials corresponding to a repartition of the mass of the path measure. We introduce these models in the abstract setting for general state spaces.

Let (E_n, ϵ_n) be a collection of measurable spaces and consider the Markov transitions $q_n(x_{n-1}, dx_n)$ from E_{n-1} to E_n and an initial probability measure $\mu \in \mathcal{P}(E_0)$. We have a Markov chain taking values $x_i \in E_i$, for simplicity we introduce the notation $x_{i:j} \coloneqq x_{i:j} \coloneqq \{x_i, \ldots, x_j\}$ and $E_{i:j} \coloneqq E_i \times \ldots \times E_j$. For any measurable function F_n on $E_{0:n}$ we have

$$\mathbb{E}_\mu\left(F_n\left(X_0, \ldots, X_n\right)\right) = \int_{E_{0:n}} F_n(x_0, \ldots, x_n)\mathbb{P}_{\mu,n}(d(x_0, \ldots, x_n)), \qquad (1.42)$$

```
Algorithm 1.4: Auxiliary particle filter
Initialise particles:
  1. For $i = 1, \dots, N$:
       (i) Sample $x_0^i \sim q(x_0)$.
      (ii) Evaluate unnormalised importance weights $w_0^i = g(y_0|x_0^i)$.
  2. Normalise importance weights and set $t = 1$.

Importance sampling:
  3. For $i = 1, \dots, N$
       (i) Sample $k_i$ according to

           $$p(k) = \frac{\widehat{p}(y_t|x_{t-1}^k)}{\sum_{j=1}^N \widehat{p}(y_t|x_{t-1}^j)}$$

      (ii) Sample $x_t^i \sim q(x_t|x_{t-1}^{k_i}, y_t)$.
     (iii) Evaluate unnormalised importance weights,

           $$w_t^i = w_{t-1}^{k_i} \frac{g(y_t|x_t^i) f(x_t^i|x_{t-1}^{k_i})}{\widehat{p}(y_t|x_{t-1}^{k_i}) q(x_t^i|x_{t-1}^{k_i}, y_t)}.$$

  4. Normalise importance weights,

           $$\tilde{w}_t^i = \frac{w_t^i}{\sum_{j=1}^N w_t^j}.$$

Resampling:
  5. Calculate ESS.
       (i) If resampling condition is met, proceed. Else, go to (8).
  6. For $i = 1, \dots, N$:
       (i) Sample $\bar{x}_{0:t}^i$ according to

           $$\widehat{p}(x_{0:t}|y_{0:t}) = \sum_{i=1}^N \delta_{x_{0:t}^i}(x_{0:t}) \tilde{w}_t^i$$

  7. Set $\{x_{0:t}^i, w_t^i\}_{i=1}^N = \{\bar{x}_t^i, \frac{1}{N}\}_{i=1}^N$.
  8. Set $t = t + 1$:
       (i) Set $t = t + 1$. If $t > T$, proceed. Else, go to (3).

Output:
  9. For $t = 0, \dots, T$:
       (i) Output $\{x_{0:t}^i, w_t^i\}_{i=1}^N$.
```

where the distribution $\mathbb{P}_{\mu,n}$ is given by

$$\mathbb{P}_{\mu,n}(d(x_0,\dots,x_n)) = \mu(dx_0)q_1(x_0,dx_1)\dots q_n(x_{n-1},dx_n). \tag{1.43}$$

Now let $G_n : E_n \to \mathbb{R}^+$ be a collection of bounded and measurable non-negative functions such that for all $n \in \mathbb{N}$

$$\mathbb{E}_\mu\left(\prod_{p=0}^{n} G_p(X_p)\right) > 0. \tag{1.44}$$

We can now define the Feynman-Kac path space models associated with the potential/transition kernel pairs (G_n, q_n).

Definition 1.4.1. *The Feynman-Kac prediction and updated path models associated with the pair (G_n, q_n), with initial distribution μ, are the sequence of path measures defined respectively by*

$$\widehat{\mathbb{Q}}_{\mu,n}(d(x_0,\dots,x_n)) \coloneqq \frac{1}{\widehat{Z}_n}\left\{\prod_{p=0}^{n-1} G_p(x_p)\right\}\mathbb{P}_{\mu,n}(d(x_0,\dots,x_n)) \tag{1.45}$$

$$\mathbb{Q}_{\mu,n}(d(x_0,\dots,x_n)) \coloneqq \frac{1}{Z_n}\left\{\prod_{p=0}^{n} G_p(x_p)\right\}\mathbb{P}_{\mu,n}(d(x_0,\dots,x_n)) \tag{1.46}$$

for any $n \in \mathbb{N}$. The normalising constants $\widehat{Z}_n$ and Z_n are given by

$$\widehat{Z}_n = \mathbb{E}_\mu\left(\prod_{p=0}^{n-1} G_p(X_p)\right), \qquad Z_n = \widehat{Z}_{n+1} = \mathbb{E}_\mu\left(\prod_{p=0}^{n} G_p(X_p)\right). \tag{1.47}$$

It is clear to see the role of the potential functions, G_n, in the above definition; they act to redistribute the probability mass associated with the Markov chain with transition kernels q_n. In a real-world inference setting the Markov chain acts as our prior while the potential functions G_n effectively give additional information to allow us to estimate the true posterior of some model.

The measures $\mathbb{Q}_{\mu,n}$ and $\widehat{\mathbb{Q}}_{\mu,n}$, defined in the above definition can also be defined for any measurable test function F_n by

$$\widehat{\mathbb{Q}}_{\mu,n}\left(F_n\right) = \frac{1}{Z_n}\mathbb{E}_\mu\left(F_n(X_0,\ldots,X_n)\prod_{p=0}^{n-1}G_p(X_p)\right), \tag{1.48}$$

$$\mathbb{Q}_{\mu,n}\left(F_n\right) = \frac{1}{\widehat{Z}_n}\mathbb{E}_\mu\left(F_n(X_0,\ldots,X_n)\prod_{p=0}^{n}G_p(X_p)\right). \tag{1.49}$$

Using this new description of the measures associated with Feynman-Kac models we can go on to define the time marginals for these models

Definition 1.4.2. *The sequence of bounded non-negative measures $\widehat{\gamma}_n$ and γ_n on E_n defined for any measurable function f_n by*

$$\widehat{\gamma}_n(f_n) = \mathbb{E}_\mu\left(f_n(X_n)\prod_{p=0}^{n-1}G_p(X_p)\right) \tag{1.50}$$

$$\gamma_n(f_n) = \mathbb{E}_\mu\left(f_n(X_n)\prod_{p=0}^{n}G_p(X_p)\right) \tag{1.51}$$

are respectively called the unnormalised prediction and the updated Feynman-Kac model associated with the potential/kernel pair (G_n, q_n). The sequence of *distributions $\widehat{\eta}_n$ and η_n on E_n defined for any measurable f_n as*

$$\widehat{\eta}_n(f_n) = \frac{\widehat{\gamma}_n(f_n)}{\widehat{\gamma}_n(1)}, \qquad \eta_n(f_n) = \frac{\gamma_n(f_n)}{\gamma_n(1)} \tag{1.52}$$

are respectively called the normalised prediction and updated Feynman-Kac model associated with the potential/kernel pair (G_n, q_n).

1.4.1 Sequential Monte Carlo as a Feynman-Kac Model

It can be useful useful to think of sequential importance sampling in terms of Feynman-Kac models. We consider a Feynman-Kac model as defined in 1.4.1 and compare this to sequential importance sampling, Algorithm 1.1.

Intuitively, the Feynman-Kac model can be thought of as a sequential change of measure between the probability law of the underlying Markov process, to some other law $\mathbb{Q}_{\mu,t}$, the modification applied to carry out this change of measure, at time t, is given by the potential function, G_t. This can also be viewed as a case of sequential importance sampling; at each t we wish to sample from the distribution $\mathbb{Q}_{\mu,t}$ by initially sampling from X_t, our underlying Markov process, and re-weighting these samples according to G_t. It's easy to deduce from 1.4.1 that we can set up a recursion for the sequence of distributions $\mathbb{Q}_t$

$$\mathbb{Q}_t(dx_{0:t}) = \frac{Z_{t-1}}{Z_t}\mathbb{Q}_{t-1}(dx_{0:t-1})q_t(x_{t-1}, dx_t)G_t(x_t). \tag{1.53}$$

A similar recursion can also be used to find the filtering distribution, $\mathbb{Q}_t(dx_t)$, which is the marginal distribution of X_t with respect to the previous states

$$\mathbb{Q}_t(dx_{t-1}, dx_t) = \frac{Z_{t-1}}{Z_t}\mathbb{Q}_{t-1}(dx_{t-1})q_t(x_{t-1}, dx_t)G_t(x_t), \tag{1.54}$$

This recursion allows us to describe sequential importance sampling in the language of Feynman-Kac models. Suppose again that we have some Feynman-Kac model with Markov process X_t on some state space E, with transitions $q_t(x_{t-1}, x_t)$, initial distribution $\mu(x_0)$ and potential functions $G_t : E \to \mathbb{R}^+$. We can proceed to recursively estimate the filtering distributions, $\mathbb{Q}_t(dx_t)$, of this model as follows.

At time 0 we sample $x_0^{1:N} \sim \mu(dx_0)$ and re-weight these particles according to G_0, so we have an approximation to $\mathbb{Q}_0(dx_0)$ as

$$\mathbb{Q}_0^N(dx_0) = \sum_{n=1}^{N} W_0^n \delta_{x_0^n}(dx_0), \qquad W_0^n = \frac{G_0(x_0^n)}{\sum_{m=1}^N G_0(x_0^m)} \tag{1.55}$$

Then for each time t, we use the recursion (1.54) to perform importance sampling at each step, using our previous approximation $\mathbb{Q}_{t-1}^N(dx_{t-1})$ in place of $\mathbb{Q}_{t-1}(dx_{t-1})$ to draw samples $x_t^{1:N}$ from a proposal distribution

$$\mathbb{Q}_{t-1}^N(dx_{t-1})q(x_{t-1}, dx_t) = \sum_{n=1}^{N} W_{t-1}^n \delta_{x_{t-1}^n}(dx_{t-1})q(x_{t-1}^n, dx_t). \tag{1.56}$$

To do this, we draw $a_t^n \sim \text{Multinomial}(W_t^{1:N})$ for each n, then sample $x_t^n \sim q(x_{t-1}^{a_t^n}, dx_t)$, these samples are then re-weighted by G_t to give an importance sampling approximation of $\mathbb{Q}_t(dx_t)$:

$$\mathbb{Q}_t^N(dx_t) = \sum_{n=1}^{N} W_t^n \delta_{x_t^n}(dx_t), \qquad W_t^n = \frac{G_t(x_t^n)}{\sum_{m=1}^{N} G_t(x_t^m)}. \tag{1.57}$$

Note that sampling the proposal in this way is equivalent to sequential importance resampling, we could skip drawing from the multinomial distribution, simply sample $x_t^n \sim q(x_{t-1}^n, dx_t)$ and propagate previous weights forward to give us

$$\mathbb{Q}_t^N(dx_t) = \sum_{n=1}^{N} W_t^n \delta_{x_t^n}(dx_t), \qquad W_t^n = W_{t-1}^n \frac{G_t(x_t^n)}{\sum_{m=1}^{N} G_t(x_t^m)} \tag{1.58}$$

as an approximation to the filtering distribution, but this would suffer from the same problems of particle degeneracy explained in the previous section.

These weights and particles can then be used to calculate estimations for the time marginals associated with the Feynman-Kac model:

$$\frac{1}{N}\sum_{n=1}^{N} \varphi(x_t^n) \approx \widehat{\eta}_t(\varphi), \qquad \frac{1}{N}\sum_{n=1}^{N} W_t^n \varphi(x_t^n) \approx \eta_t(\varphi) \tag{1.59}$$

for any measurable test function φ, where $\widehat{\eta}_t$ and η_t are defined as in 1.4.2.

This is a more abstract way of thinking of sequential Monte-Carlo methods than we introduced previously, each Feynman-Kac model generates a unique SMC algorithm. In the case of the bootstrap particle filter, the corresponding Feynman-Kac would be defined by $q(x_t \mid x_{t-1}) = p(x_t \mid x_{t-1})$ the law of the transitional prior and potential functions $G_t = g(y_t \mid x_t)$ corresponding to the likelihood of a particle given the data point at time t, then for any measurable function φ we have

$$\mathbb{E}[\varphi(X_{0:n}) \mid Y_{1:n}] = \frac{\displaystyle\int \varphi(x_{1:n}) p(x_{1:n} \mid x_0) g(y_{1:n} \mid x_{1:n}) dx_{1:n}}{\displaystyle\int p(x_{1:n} \mid x_0) g(y_{1:n} \mid x_{1:n}) dx_{1:n}} \tag{1.60}$$

$$= \frac{\displaystyle\int \varphi(x_{1:n}) p(x_{1:n} \mid x_0) \prod_{t=1}^{n} G_t(x_t) dx_{1:n}}{\displaystyle\int p(x_{1:n} \mid x_0) \prod_{t=1}^{n} G_t(x_t) dx_{1:n}} \tag{1.61}$$

$$= \frac{\mathbb{E}\left[\varphi(X_{1:n}) \displaystyle\prod_{t=1}^{n} G_t(X_t)\right]}{\mathbb{E}\left[\displaystyle\prod_{t=1}^{n} G_t(X_t)\right]}., \tag{1.62}$$

and the expectations can be approximated sequentially via the standard bootstrap filter.

1.5 Stochastic differential equations with discrete-time observations

We describe here the main problem we concern ourselves with, inference for situations where the underlying process is a continuous-time stochastic differential equation (SDE) with noisy observations at discrete time intervals.

1.5.1 Brownian motion

We introduce Brownian motion (or Wiener process), which is the main building block of many continuous time stochastic processes.

Definition 1.5.1. *Let $\{W_t\}_{t\geqslant 0}$ be a stochastic process. W_t a standard (one-dimensional) Brownian motion if it satisfies the following properties:*

1. $W_0 = 0$
2. *With probability 1, the function $t \to W_t$ is continuous in t.*
3. *For any $0 \leqslant s < t \leqslant u < v$, $W_t - W_s$ and $W_v - W_u$ are independent random variables.*
4. *For any $t > 0$, $W_{s+t} - W_s$ is normally distributed with mean 0 and variance t.*

A standard d-dimensional Brownian motion is a vector-valued stochastic process,

$$W_t = \left(W_t^1, W_t^2, \ldots, W_t^d\right),$$

where the components, W_t^i, are independent standard one-dimensional Brownian motions.

These basic properties are all we need to concern ourselves with for now, for more a rigorous and involved discussion and construction of these processes see Øksendal [1998], Szabados [2010].

1.5.2 Partially observed stochastic differential equations

We concern ourselves with problems on discretely observed non-linear SDEs. Consider the model

$$dX_t = b\left(X_t\right) dt + \sigma\left(X_t\right) dW_t, \tag{1.63}$$

where $b : \mathbb{R}^d \to \mathbb{R}^d$ is the drift term, $\sigma : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ is the diffusion matrix, W_t is a d dimensional Brownian motion and we have the initial state $X_0 = x_0 \in \mathbb{R}^d$.

We assume that the drift b and diffusion σ coefficients satisfy the conditions required to ensure the existence of a unique strong solution for (1.63). Specifically, these standard conditions require that both b and σ satisfy a global Lipschitz condition and a linear growth condition. The Lipschitz condition ensures that the coefficients do not change too rapidly, preventing explosive solutions, while the linear growth condition bounds the rate at which the coefficients can grow. For more details see Øksendal [1998], Mikosch [1998].

We have the observations Y_k

$$Y_k = X_{t_k} + N(0, \Sigma), \tag{1.64}$$

for $t_1 < \cdots < t_n = T$ and $N(\mu, \Sigma)$ is the normal distribution with mean μ and covariance Σ.

We want to compute recursively the filtering distribution of this model, i.e. the posterior distribution of X_{t_k} given the observations up to the current time,

$$p\left(X_{t_k} | Y_1, \ldots, Y_k\right). \tag{1.65}$$

1.5.3 Challenges of particle filtering in continuous time

We aim to use Sequential Monte Carlo (SMC) methods to approximate the distributions of interest and the associated normalizing constants. However, the continuous-time nature of the underlying system introduces several challenges. A primary challenge is that, unlike in the discrete-time case, the transition density, $f(x_t \mid x_{t-1})$, is not explicitly available. In particle filtering algorithms, this transition density plays a crucial role in the recursion used to update the weights (1.38). As a result, we are initially constrained to the bootstrap filter, where the weight updates rely solely on the likelihood of the observation given the proposed state of the system. This approach requires only the ability to sample from the transition density without needing its pointwise evaluation. However, as previously discussed, the bootstrap filter is often inefficient, particularly in scenarios involving high-dimensional state spaces or low observational noise, necessitating the use of more advanced methods.

1.5.4 Girsanov's theorem and diffusion bridges

In this section we outline in a simple manner some results that prove to be useful and motivating for further topics in this thesis, for a more rigorous treatment of the content found in this section see Øksendal [1998] and Delyon and Hu [2006]. Consider the SDEs

$$dX_t = b_1(X_t)dt + \sigma(X_t)dW_t, \tag{1.66}$$

$$dX_t = b_2(X_t)dt + \sigma(X_t)dW_t, \tag{1.67}$$

for $t \in [0, s]$. Denote by $\mathbb{X}$ and $\mathbb{Z}$ the measures induced on the space of continuous paths between time 0 and time s by the processes (1.66) and (1.67) respectively. A well known result in the theory of stochastic processes is that under standard regularity conditions for b_1, b_2 and σ, the measure $\mathbb{X}$ is absolutely continuous with respect to $\mathbb{Z}$ and the density of $\mathbb{X}$ with respect to $\mathbb{Z}$, known as the Radon-Nikodym derivative, is given by Girsanov's Theorem [Øksendal, 1998]:

$$\begin{aligned}\frac{d\mathbb{X}}{d\mathbb{Z}}(X) = \exp\Bigg[&\int_0^s [b_1(X_t) - b_2(X_t)]^T \sigma^{-1}(X_t) dW_t \\ &- \frac{1}{2}\int_0^s [b_1(X_t) - b_2(X_t)]^T \sigma^{-1}(X_t)[b_1(X_t) - b_2(X_t)]dt\Bigg].\end{aligned} \tag{1.68}$$

Intuitively, this Radon-Nikodym derivative (1.68), provides the ratio of the likelihood that a given sample path is sampled from $\mathbb{X}$ versus the path being sampled from $\mathbb{Z}$. This naturally lends itself to playing a role in the importance weight of an importance sampler, allowing us to evaluate expectations under $\mathbb{X}$ by sampling from $\mathbb{Z}$ and re-weighting the samples according to the Radon-Nikodym derivative (1.68).

A frequently studied problem is that of sampling from a stochastic process X_t, as defined in (1.66), conditioned so that $X_0 = x$ and $X_s = y$, such processes are commonly referred to in the literature as "diffusion bridges". The addition of this conditioning introduces complex dependencies meaning it is generally not possible to calculate the transition densities required for direct simulation of sample paths. However, by using a version of (1.68) for diffusion bridges we can instead sample from a simpler process to generate proposals for the target process and re-weight these proposals according to the Radon-Nikodym derivative. Again let $\mathbb{X}$ and $\mathbb{Z}$ denote the measures induced on the space of continuous paths between time 0 and time s by the processes (1.66) and (1.67).

Applying Bayes theorem to $\frac{d\mathbb{X}}{d\mathbb{Z}}(X \mid X_0 = x, X_s = y)$, we obtain

$$\frac{d\mathbb{X}}{d\mathbb{Z}}(X|X_0 = x, X_s = y) = \frac{\mathbb{X}(X_s = y|X)\mathbb{X}(dX)/\mathbb{X}(X_s \in dy)}{\mathbb{Z}(X_s = y|X)\mathbb{Z}(dX)/\mathbb{Z}(X_s \in dy)}, \tag{1.69}$$

where, for simplicity, the conditioning on $X_0 = x$ is omitted in the notation but is easy to impose as required. As we are considering constrained paths, by definition $\mathbb{X}(X_s = y|X) = 1$ and $\mathbb{Z}(X_s = y|X) = 1$ and from the Girsanov theorem we have that the density $\frac{\mathbb{X}(dX)}{\mathbb{Z}(dX)}$ is given by (1.68). Denoting (1.68) as $G(X)$ we have

$$\frac{d\mathbb{X}}{d\mathbb{Z}}(X|X_0 = x, X_s = y) = G(X)\frac{\mathbb{Z}(X_s \in dy \mid X_0 = x)}{\mathbb{X}(X_s \in dy \mid X_0 = x)}. \tag{1.70}$$

Assuming (1.66) and (1.67) have analytically available transition densities $p_s(y \mid x)$ and $q_s(y \mid x)$ respectively, we can write

$$\frac{d\mathbb{X}}{d\mathbb{Z}}(X|X_0 = x, X_s = y) = G(X)\frac{q_s(y \mid x)}{p_s(y \mid x)}. \tag{1.71}$$

Of note here is the fact that $q_s(x, y)$ and $p_s(x, y)$ will be constant across all samples from the proposal process and therefore $\frac{q_s(y|x)}{p_s(y|x)}$ acts as a normalising constant and does not need to be evaluated pointwise, this allows the use (e.g. as sample weight in an importance sampler or as part of acceptance probabilities in a Metropolis-Hastings framework) of (1.71) without the need for explicit calculation of the transition densities of the unconditional processes (1.66) and (1.67).

It still remains a challenge to find a proposal process that is straightforward to sample from in the case where the target distribution is that of a diffusion bridge. One commonly used example in the literature is the so called Brownian bridge (see e.g. Durham and Gallant [2002]). Suppose we wish to sample from a diffusion bridge

$$dX_t = b(X_t)dt + \sigma(X_t)dW_t, \qquad X_0 = x, X_s = y, \tag{1.72}$$

and denote by $\mathbb{X}_{x,y}$ the measure induced on the space of continuous paths between time 0 and time s by the process (1.72). The Brownian bridge is a process

$$dX_t = \tilde{b}(X_t)dt + \sigma(X_t)dW_t, \qquad X_0 = x, \tag{1.73}$$

where

$$\tilde{b}(X_t) = \frac{y - X_t}{s - t}. \tag{1.74}$$

The specific form of the drift (1.74), ensures that the process satisfies the condition $X_s = y$ and is straightforward to sample from. Denote by $\mathbb{W}_{x,y}$ the measure induced on the space of continuous paths between time 0 and time s by the process (1.73), then we have

$$\frac{d\mathbb{X}_{x,y}}{d\mathbb{W}_{x,y}}(X) = \tilde{G}(X)\frac{w_s(y \mid x)}{p_s(y \mid x)} \tag{1.75}$$

$$\propto \tilde{G}(X), \tag{1.76}$$

where $\tilde{G}(X)$ represents the Radon-Nikodym derivate defined in (1.68) with $b_1 = b$ and $b_2 = \tilde{b}$ and $w_s(y \mid x)$ and $p_s(y \mid x)$ are the transition densities of the unconditioned versions of the processes (1.72) and (1.73) respectively.

Chapter 2

Beyond the bootstrap filter

In this section we highlight some current approaches from the literature that provide more sophisticated methods than bootstrap for particle filtering on SDEs.

2.1 Intermediate weighting

Del Moral and Murray [2015] propose methods which introduce a series of intermediate times between observations, at which weighting and resampling occur, in an attempt to guide the proposal toward the observation. They expect these methods to be particularly effective when the observations are highly informative.

Consider the situation in which we have a continuous time Markov process, $X_t \in \mathbb{R}^d$, satisfying some SDE like (1.63). For a sequence of times $t_0, \cdots, t_n$ write $X_{0:n} := \{X_{t_0}, \cdots, X_{t_n}\}$. The process is observed at the final time, t_n, via some random variable, $Y_n \sim p(y_n|x_n)$, and we have a prior distribution over the initial state of the system, $X_0 \sim p(x_0)$. The method seeks to simulate $X_{0:n} \sim p(x_{0:n}|y_n)$ and is underpinned by the following [Del Moral and Murray, 2015]:

Proposition 2.1.1. *For any bounded function φ on $\mathbb{R}^{d(n+1)}$, we have*

$$\mathbb{E}\left[\varphi(X_{0:n})|y_n\right] = \frac{\mathbb{E}\left[\varphi(X_{0:n})p(X_0)p(y_n|X_n)\frac{r(y_n|X_0)}{r(y_n|X_n)}\prod_{k=1}^{n} J_k(X_{k-1:k})\right]}{\mathbb{E}\left[p(X_0)p(y_n|X_n)\frac{r(y_n|X_0)}{r(y_n|X_n)}\prod_{k=1}^{n} J_k(X_{k-1:k})\right]}$$

with the weight functions

$$J_k(x_{k-1:k}) \coloneqq \frac{r(y_n|x_k)}{r(y_n|x_{k-1})}$$

and chosen positive functions $r(y_t|x_k)$ for $k = 0, \ldots, N$.

Proof. First notice that

$$\begin{aligned}\mathbb{E}\left[p(y_n|X_n)\right] &= \int p(y_n|x_n)p(x_n)dx_n \\ &= p(y_n)\end{aligned}$$

then we have

$$\begin{aligned}\mathbb{E}\left[\varphi(X_{0:n})|y_n\right] &= \int \varphi(x_{0:n})p(x_{0:n}|y_n)dx_{0:n} \\ &= \int \frac{\varphi(x_{0:n})p(y_n|x_{0:n})p(x_{0:n})}{p(y_n)}dx_{0:n} \\ &= \frac{\int \varphi(x_{0:n})p(y_n|x_n)p(x_{0:n})}{p(y_n)}dx_{0:n} \\ &= \frac{\mathbb{E}\left[\varphi(X_{0:n})p(y_n|X_n)\right]}{\mathbb{E}\left[p(y_n|X_n)\right]}\end{aligned}$$

Introduce the weighting functions and notice

$$\frac{r(y_n|x_0)}{r(y_n|x_n)}\prod_{k=1}^{n} J_k(x_{k-1:k}) = 1.$$

We then have

$$\mathbb{E}\left[\varphi(X_{0:n})|y_n\right] = \frac{\mathbb{E}\left[\varphi(X_{0:n})p(X_0)p(y_n|X_n)\dfrac{r(y_n|X_0)}{r(y_n|X_n)}\displaystyle\prod_{k=1}^{n} J_k(X_{k-1:k})\right]}{\mathbb{E}\left[p(X_0)p(y_n|X_n)\dfrac{r(y_n|X_0)}{r(y_n|X_n)}\displaystyle\prod_{k=1}^{n} J_k(X_{k-1:k})\right]}.$$

□

The recursive nature of the weight functions $J_k(x_{k-1:k})$ in proposition 2.1.1 suggests the use of a particle filter with intermediate steps between observation times, repeatedly propagating particles forward, weighting and resampling to help guide the proposal toward the next state.

The methodology is perhaps best understood via the Feynman-Kac framework. Recall the Feynman-Kac model for the standard particle filter outlined in (1.60), (1.61) and (1.62). The intermediate weighting method can be seen as a modification of this, where we have a sequence of intermediate times, $t_0, \cdots, t_n$, and only a single data point y_n at the final time, t_n. The expectation of interest for a measurable function φ is then

$$\mathbb{E}\left[\varphi(X_{0:n}) \mid y_n\right] = \frac{\displaystyle\int \varphi(x_{0:n})p(x_{1:n} \mid x_0)p(x_0)p(y_n \mid x_n)dx_{0:n}}{\displaystyle\int p(x_{1:n} \mid x_0)p(x_0)p(y_n \mid x_n)dx_{0:n}}, \tag{2.1}$$

and by proposition 2.1.1 we obtain

$$\mathbb{E}\left[\varphi(X_{0:n})|y_n\right] = \frac{\mathbb{E}\left[\varphi(X_{0:n})p(X_0)p(y_n|X_n)\frac{r(y_n|X_0)}{r(y_n|X_n)}\prod_{k=1}^{n}\frac{r(y_n|X_k)}{r(y_n|X_{k-1})}\right]}{\mathbb{E}\left[p(X_0)p(y_n|X_n)\frac{r(y_n|X_0)}{r(y_n|X_n)}\prod_{k=1}^{n}\frac{r(y_n|X_k)}{r(y_n|X_{k-1})}\right]}, \tag{2.2}$$

for some positive functions $r(y_n|x_k)$. We can write (2.2) as

$$\mathbb{E}\left[\varphi(X_{0:n})|y_n\right] = \frac{\mathbb{E}\left[\varphi(X_{0:n})p(X_0)\prod_{k=1}^{n}G_k(X_k, X_{k-1})\right]}{\mathbb{E}\left[p(X_0)\prod_{k=1}^{n}G_k(X_k, X_{k-1})\right]}, \tag{2.3}$$

where

$$\begin{aligned} G_1(x_1, x_0) &:= \frac{r(y_n \mid x_1)}{r(y_n \mid x_0)} r(y_n \mid x_0) \\ &= r(y_n \mid x_1), \\ G_k(x_k, x_{k-1}) &:= \frac{r(y_n \mid x_k)}{r(y_n \mid x_{k-1})}, \qquad k = 2, \cdots, n-1, \\ G_n(x_n, x_{n-1}) &:= \frac{r(y_n \mid x_n)}{r(y_n \mid x_{n-1})} \frac{p(y_n \mid x_n)}{r(y_n \mid x_n)} \\ &= \frac{p(y_n \mid x_n)}{r(y_n \mid x_{n-1})}. \end{aligned} \tag{2.4}$$

Therefore, we have a Feynman-Kac model with transition kernel $p(x_{k+1} \mid x_k)$ and incremental weights $G_k(X_k, X_{k-1})$, this differs from the model of the standard particle filter in that the weights are not just a function of the current position of the particle but a function of the current and previous positions. It's now straightforward to construct a particle filter to estimate the expectation in (2.1), this can be seen in Algorithm 2.1.

```
Algorithm 2.1: Particle filtering with intermediate weighting
Initialise particles:
  1. For $i = 1, \dots, N$:
       (i) Sample $x_0^i \sim p(x_0)$.
      (ii) Set all weights $w_0^i = \frac{1}{N}$.

Importance sampling:
  3. For $k = 1, \dots, n$:
       (i) Sample $x_k^i \sim p(x_k | x_{k-1}^i)$.
      (ii) Evaluate unnormalised importance weights,
               $w_k^i = w_{k-1}^i G_k(x_k^i, x_{k-1}^i).$
     (iii) Calculate ESS
             If resampling is triggered:
                 (a) For $i = 1, \dots, N$, sample $\bar{x}_k^i$ according to
                         $p(x_k^i) = \sum_{i=1}^{N} \delta_{x_k^i}(x_k) w_k^i$
                 (b) Set $\{x_k^i, w_k^i\}_{i=1}^N = \{\bar{x}_k^i, \frac{1}{N}\}_{i=1}^N$.
             Else:
                 (a) Normalise importance weights,
                         $w_k^i = \frac{w_k^i}{\sum_{j=1}^N w_k^j}$.
  4. Set $t = t + 1$. If $t > T$, proceed. Else, go to (3).
```

Like the bootstrap filter, this algorithm requires only that we can simulate from the transition density $p(x_k|x_{k-1})$ and not that it can be evaluated pointwise. Other than this, the method places few assumptions on the underlying SDE and can be applied to a wide range of problems.

The choice of the weighting function, $r(y_n|x_k)$, is key to the performance of the algorithm. For good performance we would like $r(y_n|x_k) \approx p(y_n|x_k)$. It is suggested that a useful approach is often to fit a Gaussian process to the observed data and construct weight functions based around that, see Del Moral and Murray [2015] for details.

2.2 Random weight particle filter

The approach presented in Fearnhead et al. [2008] is for d-dimensional SDEs of the form

$$dZ_s = \alpha(Z_s)ds + dW_s, \tag{2.5}$$

with observations at discrete time steps $t_0, \cdots, t_n$, related to the signal by a known density $g(y_j|z_{t_j})$. Some further assumptions on (2.5) are also required:

1. α is continuously differentiable in all arguments.
2. There exists some $A : \mathbb{R}^d \to \mathbb{R}$ such that $\alpha(z) = \nabla A(z)$.
3. $\phi(z)$ is bounded below by some $l \in \mathbb{R}$, where

$$\phi(z) := \frac{||\alpha(z)||^2 + \Delta A(z)}{2}. \tag{2.6}$$

Where Δ is the Laplacian operator and $||\cdot||$ is the Euclidean norm. To apply the methods to a more general SDE of the form (1.63), we require that there is an explicit transformation, $\eta(X_s) = Z_s$, where Z solves an SDE of form (2.5) and the conditions specified above are satisfied. The density of the observations then becomes $g(y_j|\eta^{-1}((Z_{t_j}))$. Finding such a transformation is fairly straightforward under mild conditions when $d = 1$, but is much harder, or even impossible, in higher dimensions [Fearnhead et al., 2008].

The approach follows a framework similar to the auxiliary particle filter, defined in section 1.3.2. We have a proposal density of the form

$$\sum_{i=1}^{N} \beta_{j-1}^{i} q(z_{t_j} | z_{t_{j-1}}^{i}, y_j); \tag{2.7}$$

to sample a new particle at time t_j, we first sample $k_i \in \{1, \cdots, N\}$ where $p(k) = \beta_{j-1}^{k}$, then simulate a new state $z_{t_j}^{i} \sim q(z_{t_j} | z_{t_{j-1}}^{k_i}, y_j)$. We are free to choose β_{j-1}^{k} and $q(z_{t_j} | z_{t_{j-1}}, y_j)$. The pairs $\{(z_{t_{j-1}}^{k_i}, z_{t_j}^{i})\}_{i=1}^{N}$ are then assigned weights using the recursion

$$w_j^i \propto w_{j-1}^{k_i} \frac{g(y_j | z_{t_j}^i) p(z_{t_j}^i | z_{t_{j-1}}^{k_i})}{\beta_{j-1}^{k_i} q(z_{t_j}^i | z_{t_{j-1}}^{k_i}, y_j)}, \tag{2.8}$$

For most SDEs of interest, the transition density $p(z_{t_j} | z_{t_{j-1}})$ is intractable. The random weight particle filter aims to get around this problem by replacing the weight, (2.8), with an unbiased estimator.

2.2.1 Weight simulation

Though the transition of density of a general SDE is usually intractable, there is an expression available in the case of (2.5) [Papaspiliopoulos, 2011]. Denote by $\mathbb{Z}_{z_t}$ the density on the space of continuous paths between time 0 and t induced by the diffusion bridge with SDE (2.5) and $Z_0 = z_0$, $Z_t = z_t$. Likewise, denote by $\mathbb{W}_{z_t}$ the equivalent but where the underlying SDE is simply a Brownian motion. From the Girsanov formula for diffusion bridges (1.71) we have

$$\frac{d\mathbb{Z}_{z_t}}{d\mathbb{W}_{z_t}}(Z) = \frac{d\mathbb{Z}}{d\mathbb{W}}(Z)\frac{\mathcal{N}_t(z_t - z_0)}{p(z_t \mid z_0)}, \tag{2.9}$$

where $p(z_t \mid z_0)$ is the transition density of the SDE (2.5) and $\mathcal{N}_t(u)$ is the density of a d-dimensional normal distribution with mean 0 and variance tI_d evaluated at $u \in \mathbb{R}^d$ (the transition density of a d-dimensional Brownian motion). $\mathbb{Z}$ and $\mathbb{W}$ are the densities corresponding to the unconditioned SDE (2.5) and Brownian motion respectively. Rearranging (2.9) and taking expectations with respect to $\mathbb{W}_{z_t}$ gives

$$p(z_t \mid z_0) = \mathcal{N}_t(z_t - z_0)\mathbb{E}\left[\frac{d\mathbb{Z}}{d\mathbb{W}}(Z)\right]. \tag{2.10}$$

The Girsanov formula for unconditional SDEs (1.68) gives that

$$\frac{d\mathbb{Z}}{d\mathbb{W}}(Z) = \exp\left\{\int_0^t \alpha(Z_s)dW_s - \frac{1}{2}\int_0^t ||\alpha(Z_s)||^2 ds\right\}. \tag{2.11}$$

Recalling that $\alpha(z) = \nabla A(z)$, we can rewrite (2.11) as

$$\frac{d\mathbb{Z}}{d\mathbb{W}}(Z) = \exp\left\{\int_0^t \nabla A(Z_s)dW_s - \frac{1}{2}\int_0^t ||\nabla A(Z_s)||^2 ds\right\}. \tag{2.12}$$

We now focus on the integral with respect to W_s in (2.12). As A is twice continuously differentiable, by Ito's Lemma [Øksendal, 1998] we know that

$$dA(Z_s) = \nabla A(Z_s)dZ_s + \frac{1}{2}\Delta A(Z_s)ds. \tag{2.13}$$

From our initial SDE (2.5) we have that $dZ_s = \nabla A(Z_s)ds + dW_s$, substitute this into (2.13) and rearrange to get (using the convention that $dW_s dW_s = ds$)

$$\nabla A(Z_s)dW_s = dA(Z_s) - \left(||\nabla A(Z_s)||^2 - \frac{1}{2}\Delta A(Z_s) \right) ds, \tag{2.14}$$

therefore

$$\int_0^t \nabla A(Z_s)dW_s = A(Z_t) - A(Z_0) - \int_0^t ||\nabla A(Z_s)||^2 ds - \int_0^t \frac{1}{2}\Delta A(Z_s)ds. \tag{2.15}$$

Combining (2.12) and (2.15) gives

$$\begin{aligned} \frac{d\mathbb{Z}}{d\mathbb{W}}(Z) &= \exp\left\{ A(Z_t) - A(Z_0) - \int_0^t ||\nabla A(Z_s)||^2 - \frac{1}{2}\Delta A(Z_s)ds \right\} \\ &= \exp\{A(Z_t) - A(Z_0)\} \exp\left\{ -\int_0^t \phi(Z_s)ds \right\}, \end{aligned} \tag{2.16}$$

where $\phi(z)$ is defined in (2.6). Finally, combining (2.10) and (2.16) gives the transition density of the SDE (2.5) as

$$p(z_t|z_0) = \mathcal{N}_t(z_t - z_0) \exp\{A(z_t) - A(z_0)\} \mathbb{E}\left[\exp\left\{ -\int_0^t \phi(W_s)ds \right\} \right], \tag{2.17}$$

where the expectation is taken with respect to a Brownian bridge, W_s, with $W_0 = z_0$ and $W_t = z_t$.

Combining expression (2.17) with (2.8) give us a weight of the form

$$w_j^i \propto h_j(z_{t_{j-1}}^{k_i}, z_{t_j}^i, y_j)\mu_\phi(z_{t_{j-1}}^{k_i}, z_{t_j}^i, t_{j-1}, t_j), \tag{2.18}$$

where

$$h_j(z^{k_i}_{t_{j-1}}, z^i_{t_j}, y_j) = w^{k_i}_{j-1} \frac{g(y_j|z^i_{t_j})\mathcal{N}_{t_j - t_{j-1}}(z^i_{t_j} - z^{k_i}_{t_{j-1}}) \exp\{A(z^i_{t_j}) - A(z^{k_i}_{t_{j-1}})\}}{\beta^{k_i}_{j-1} q(z^i_{t_j}|z^{k_i}_{t_{j-1}}, y_j)}, \tag{2.19}$$

and

$$\mu_\phi(z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j) := \mathbb{E}\left[\exp\left\{-\int_{t_{j-1}}^{t_j} \phi(W_s)ds\right\}\right], \tag{2.20}$$

where the expectation is with respect to a Brownian bridge W_s, with $W_{t_{j-1}} = z^{k_i}_{t_{j-1}}$ and $W_{t_j} = z^i_{t_j}$.

Typically, the expectation in (2.20) is intractable. We would therefore like to replace $\mu_\phi(z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j)$ with an unbiased estimator in our calculation of the weights (2.18). The approach of Fearnhead et al. [2008] is to construct auxiliary random variables, V, and an easy to evaluate function, $r(v_j, z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j) \geqslant 0$, such that

$$\mathbb{E}\left[r(V, z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j)\right] = \mu_\phi(z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j), \tag{2.21}$$

and the auxiliary variables are sampled via $V \sim Q(v|z^{k_i}_{t_{j-1}}, z^i_{t_j}, t_{j-1}, t_j)$, for some appropriate distribution Q. If we can construct Q and r such that (2.21) holds then we can replace μ_ϕ in (2.18) with an unbiased estimator and therefore construct a particle filter that targets the correct filtering distribution. This gives rise to the *random weight particle filter*, as outlined in Algorithm 2.2.

Algorithm 2.2: Random weight particle filter

```
1. For $i = 1, \cdots, N$, sample $z_0^i \sim p(z_0)$ and set $w_0^i = \frac{1}{N}$. Set $j = 1$.
2. Calculate ESS of $\{\beta_{j-1}^i\}_{i=1}^N$.
     If resampling condition is met:
       For $i = 1, \cdots, N$:
         (i) Sample $k_i$ from $\{1, \cdots, N\}$ with $p(k) \propto \beta_{j-1}^k$
        (ii) Set $\delta_j^i = 1$.
     Else:
       For $i = 1, \cdots, N$:
         (i) Set $k_i = i, \delta_j^i = \beta_{j-1}^i$
3. For $i = 1, \cdots, N$:
      (i) Sample next state $z_{t_j}^i \sim q(z_{t_j} | z_{t_{j-1}}, y_j)$
     (ii) Sample auxiliary variables $v_j^i \sim Q(v_j | z_{t_{j-1}}^{k_i}, z_{t_j}^i, t_{j-1}, t_j)$
4. For $i = 1, \cdots, N$:
      (i) Calculate particle weight:

          $$w_j^i \propto \delta_j^i h_j(z_{t_{j-1}}^{k_i}, z_{t_j}^i, y_j) r(v_j^i, z_{t_{j-1}}^{k_i}, z_{t_j}^i, t_{j-1}, t_j)$$

5. Set $j = j + 1$. If $j > n$, proceed. Else, go to (2).
6. For $j = 1, \cdots, N$:
      (i) Output particles $\{z_{t_j}^i, w_j^i\}_{i=1}^N$
```

2.2.2 Generalised Poisson Estimators

To carry out the random weight particle filter we need an unbiased estimator of (2.20). Fearnhead et al. [2008] builds on the work of Beskos et al. [2006] to construct a family of such estimators with desirable properties, the so called *Generalised Poisson Estimator* (GPE).

This is introduced in a general context of simulating an unbiased estimator of

$$\mathbb{E}\left[\exp\left\{-\int_0^t g(W_s)ds\right\}\right], \tag{2.22}$$

where the expectation is taken with respect to a d-dimensional Brownian bridge and g is an arbitrary continuous function on $\mathbb{R}^d$. It is assumed that $W_0 = x_1$ and $W_t = x_2$ for arbitrary $x_1, x_2 \in \mathbb{R}^d$ and that $t > 0$. It is also noted that the time-homogenous nature of Brownian bridges ensures the method extends to the case where the integration limits are changed to t_0 and $t_0 + t$ for any $t > 0$.

The GPE builds on the Poisson Estimator proposed in Wagner [1988]. For the case where g is bounded. The Poisson Estimator is given by

$$e^{(\lambda-c)t}\lambda^{-\kappa}\prod_{j=1}^{\kappa}\left[c-g(W_{\psi_j})\right], \tag{2.23}$$

where κ is a Poisson(λt) random variable and ψ_j are distributed uniformly on $[0,t]$ for all j. This estimator is unbiased, to see this we first take the expectation of (2.23) with respect to κ to see that

$$\begin{aligned}\mathbb{E}\left[e^{(\lambda-c)t}\lambda^{-\kappa}\prod_{j=1}^{\kappa}\left[c-g(W_{\psi_j})\right]\right] &= \mathbb{E}\left[\sum_{k=0}^{\infty}e^{(\lambda-c)t}\lambda^{-k}\prod_{j=1}^{k}\left[c-g(W_{\psi_j})\right]\frac{\lambda^k t^k e^{-\lambda t}}{k!}\right] \\ &= \mathbb{E}\left[\sum_{k=0}^{\infty}e^{-ct}\frac{1}{k!}\prod_{j=1}^{k}\left[ct-g(W_{\psi_j})t\right]\right].\end{aligned} \tag{2.24}$$

As we have $\psi_j \sim U[0,t]$, $g(W_{\psi_j})t$ is an unbiased estimator of $\int_0^t g(W_s)ds$ and

$$\begin{aligned}
\mathbb{E}\left[\sum_{k=0}^{\infty} e^{-ct}\frac{1}{k!}\prod_{j=1}^{k}\left[ct - g(W_{\psi_j})t\right]\right] &= \mathbb{E}\left[\sum_{k=0}^{\infty} e^{-ct}\frac{1}{k!}\mathbb{E}\left[\prod_{j=1}^{k}\left[ct - g(W_{\psi_j})t\right]\right]\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{\infty} e^{-ct}\frac{1}{k!}\prod_{j=1}^{k}\left[ct - \int_0^t g(W_s)ds\right]\right] \\
&= \mathbb{E}\left[\sum_{k=0}^{\infty} e^{-ct}\frac{1}{k!}\left[ct - \int_0^t g(W_s)ds\right]^k\right] \\
&= \mathbb{E}\left[\exp\left\{-\int_0^t g(W_s)ds\right\}\right].
\end{aligned} \tag{2.25}$$

The estimator (2.23) is not sure to have finite variance for unbounded g, it may also return negative estimates. The methods of Fearnhead et al. [2008] generalise (2.23), allowing c and λ to depend on W and allowing κ to be a general discrete distribution.

This generalisation requires the ability to simulate random variables L_W and U_W such that

$$L_W \leqslant g(W_s) \leqslant U_W, \qquad \text{for all } s \in [0,t], \tag{2.26}$$

along with the ability to simulate W_s for all s, conditional on L_W and U_W in (2.26). An efficient algorithm for performing these simulations can be found in Beskos et al. [2008a].

Assuming L_W and U_W satisfy the condition (2.26), introduce $\{\psi_j\}$, a sequence of uniform random variables on $[0,t]$ and notice that we can rewrite

(2.22) as

$$\mathbb{E}\left[\exp\left\{-\int_0^t g(W_s)ds\right\}\right] = \mathbb{E}\left[e^{-U_W t}\exp\left\{\int_0^t (U_W - g(W_s))ds\right\}\right] \tag{2.27}$$

$$= \mathbb{E}\left[e^{-U_W t}\sum_{k=0}^{\infty}\frac{1}{k!}\left(\int_0^t (U_W - g(W_s))ds\right)^k\right] \tag{2.28}$$

Notice that if $\psi \sim U[0,t]$, then $t(U_W - g(W_\psi))$ is an unbiased estimator of $\int_0^t U_W - g(W_s)ds$, conditioned on U_W, L_W. So (2.28) becomes

$$\mathbb{E}\left[e^{-U_W t}\sum_{k=0}^{\infty}\frac{1}{k!}\left(\mathbb{E}\left[t(U_W - g(W_\psi))|U_W, L_W\right]\right)^k\right] \tag{2.29}$$

$$= \mathbb{E}\left[e^{-U_W t}\sum_{k=0}^{\infty}\frac{t^k}{k!}\mathbb{E}\left[\prod_{j=1}^{k}(U_W - g(W_{\psi_j}))|U_W, L_W\right]\right] \tag{2.30}$$

where the equality of the second line in the above comes from the fact that $\{\psi_j\}$ are i.i.d. and drawn from $U[0,t]$. We can apply Fubini's theorem to (2.30) to swap the order of the infinite summation and expectation, giving

$$\mathbb{E}\left[e^{-U_W t}\mathbb{E}\left[\sum_{k=0}^{\infty}\frac{t^k}{k!}\prod_{j=1}^{k}(U_W - g(W_{\psi_j}))|U_W, L_W\right]\right] \tag{2.31}$$

$$= \mathbb{E}\left[e^{-U_W t}\frac{t^\kappa}{\kappa! p(\kappa|U_W, L_W)}\prod_{j=1}^{\kappa}(U_W - g(W_{\psi_j}))\right], \tag{2.32}$$

where κ is a discrete random variable with probabilities $p(k|U_W, L_W)$.

By specifying $p(k|U_W, L_W)$ we can derive various different estimators of

(2.22), this family of estimators is the GPE and is of the form

$$e^{-U_W t}\frac{t^\kappa}{\kappa! p(\kappa|U_W, L_W)}\prod_{j=1}^{\kappa}(U_W - g(W_{\psi_j})). \tag{2.33}$$

2.3 Using Girsanov's theorem with a scaled proposal

Särkkä et al. [2008] introduces a method that uses Girsanov's theorem to construct likelihood ratios between the densities of two stochastic differential equations (SDEs). The authors show that when working with original and proposal SDEs that have invertible but different diffusion matrices, it is possible to appropriately scale the proposal process to match the diffusion matrix of the original process. This adjustment guarantees absolute continuity between the densities of the original and the scaled process, allowing for the establishment of a Radon-Nikodym derivative. This insight expands the range of potential proposal processes that can be applied within a particle filtering framework.

The method is presented in the context of estimating the filtering distribution of an SDE of the form

$$dX_t = b_1(X_t)dt + \sigma_1(t)dW_t, \tag{2.34}$$

with observations at discrete time steps $t_0, \cdots, t_n$, related to the signal by a known density $g(y_j|x_{t_j})$ and with $\sigma_1(t)$ invertible for all t. Assume that we have constructed a proposal process that approximates the filtering distribution of (2.34) with an SDE of the form

$$dZ_t = b_2(Z_t)dt + \sigma_2(t)dW_t, \tag{2.35}$$

where $\sigma_2(t)$ is also invertible for all t. Define the process Z_t^* as

$$dZ_t^* = \sigma_1(t)\sigma_2^{-1}(t)dZ_t \tag{2.36}$$

and denote by $\mathbb{Z}^*$ and $\mathbb{X}$ the densities induced on the space of continuous paths between time 0 and time s by (2.36) and (2.34). Then $\mathbb{Z}^*$ is absolutely continuous with respect to $\mathbb{X}$ and the Radon-Nikodym derivative is given by

$$\begin{aligned}\frac{d\mathbb{X}}{d\mathbb{Z}^*}(Z) = \exp\Bigg[\int_0^s & B(Z_t,t)^T\sigma_1^{-T}(t)dW_t \\ & -\frac{1}{2}\int_0^s B(Z_t,t)^T[\sigma_1(t)\sigma_1(t)^T]^{-1}B(Z_t,t)dt\Bigg],\end{aligned} \tag{2.37}$$

where $B(Z_t,t) = b_1(Z_t^*) - \sigma_1(t)\sigma_2^{-1}(t)b_2(Z_t)$. We omit the technical details here for brevity, but a proof can be found in the appendix of Särkkä et al. [2008].

The method of Särkkä et al. [2008] is to use the Radon-Nikodym derivative (2.37) to construct a particle filter that targets the filtering distribution of (2.34) using the proposal process (2.35). Weight updates are calculated as

$$w_j^i = w_{j-1}^i G_{x_{j-1}^i}(Z^i, y_j)g(y_j|x_{t_j}^i), \tag{2.38}$$

Algorithm 2.3: Particle filter with transformed proposal

```
1. For $i = 1, \cdots, N$, sample $x_0^i \sim p(x_0)$ and set $w_0^i = \frac{1}{N}$. Set $j = 1$.
2. For $i = 1, \cdots, N$,
     (i) Generate proposals on $[t_{j-1}, t_j]$:
           $dZ_t^i = b_2(Z_t^i)dt + \sigma_2(t)dW_t, \qquad Z_{t_{j-1}}^i = x_{t_{j-1}}^i$
           $dZ_t^{*,i} = \sigma_1(t)\sigma_2^{-1}(t)dZ_t^i, \qquad Z_{t_{j-1}}^{*,i} = x_{t_{j-1}}^i$
    (ii) Update importance weights:
           $$w_j^i = w_{j-1}^i G_{x_{t_{j-1}}^i}(Z^i) g(y_j | x_{t_j}^i)$$
3. Normalise importance weights.
4. Calculate ESS of $\{w_j^i\}_{i=1}^N$:
     (i) If resampling condition is met, resample particles and set
         $w_j^i = \frac{1}{N}$.
5. Set $j = j + 1$, if $j > n$ proceed, else go to (2).
6. For $j = 1, \cdots, n$:
     (i) Output particles $\{x_j^i, w_j^i\}_{i=1}^N$
```

where $G_{x_{t_{j-1}}^i}(Z^i)$ is the likelihood ratio as defined in (2.37) between times t_{j-1} and t_j. The method is outlined in Algorithm 2.3.

2.3.1 Proposal process via the extended Kalman filter

A suggested approach to generating a proposal process involves using the continuous discrete extended Kalman filter (CDEKF) (see e.g. Jazwinski [2007]). Here, we provide a brief outline of how such a proposal process can be created.

We wish to generate a proposal process for filtering of a SDE of the form of 2.34, with observations

$$y_k = h(x_{t_k}) + \eta, \tag{2.39}$$

where h is a differentiable function and $\eta \sim N(0, \Sigma)$. Suppose that we are at time t_{k-1}, we have the initial condition $X_{t_{k-1}} = x_{t_{k-1}}$ and we wish to generate the proposal process up to time t_k. The CDEKF estimates a Gaussian posterior distribution of the state of the system at time t_k and is made up of two stages:

- A data-blind prediction step, giving initial estimates for the mean and covariance of the state of the system at time t_k.
- An update step to take into account the information given by the observed data point.

To carry out the prediction step, start from initial conditions $X_{t_{k-1}} = x_{t_{k-1}}$ and $P(t_{k-1}) = 0$ and integrate between t_{k-1} and t_k the differential equations

$$\begin{aligned} \frac{dX_t}{dt} &= b_1(X_t) \\ \frac{dP(t)}{dt} &= F(t)P(t) + P(t)F(t)^T + \sigma_1(t), \end{aligned} \tag{2.40}$$

where F is the Jacobian of b_1 with respect to the state of the system. This gives initial estimates of the mean and covariance which we denote by $\hat{x}_{k|k-1}$ and $\hat{P}_{k|k-1}$ respectively. These estimates are then updated via

$$\begin{aligned} K_k &= \hat{P}_{k|k-1}H_k^T(H_k\hat{P}_{k|k-1}H_k^T + \Sigma)^{-1}, \\ \hat{x}_k &= \hat{x}_{k|k-1} + K_k(y_k - h(\hat{x}_{k|k-1}), \\ \hat{P}_k &= (I_d - K_kH_k)\hat{P}_{k|k-1}, \end{aligned} \tag{2.41}$$

where H_k is the Jacobian of $h(\hat{x}_{k|k-1})$ and I_d is an identity matrix of appropriate dimension.

We now have an estimate of how the system is distributed at time t_k as a

Gaussian distribution with mean $\hat{x}_k$ and covariance $\hat{P}_k$. A proposal process that satisfies this is given by

$$dX_t = \frac{(\hat{x}_k - x_{t_{k-1}})}{\Delta t} dt + \frac{1}{\sqrt{\Delta t}} \hat{P}_k^{1/2} dW_t. \tag{2.42}$$

Chapter 3

Particle filtering with tempering and mutation for partially observed SDEs

We aim to develop an algorithm that allows for more stability in the particle weights and is more robust than the standard bootstrap filter. In particular, we focus on developing methods which are robust to the dimension of the SDE and perform well in the small noise setting.

The primary challenge in filtering high-dimensional systems with informative observations is the rapid degeneracy of particle weights, a problem where the bootstrap filter often fails. While the components of our proposed algorithm have appeared individually in prior work (see e.g. Andrieu et al. [2010], Doucet et al. [2000], Del Moral et al. [2006]), the novelty of our approach lies in the combination of these techniques to address this challenge.

The algorithm we propose has three key elements:

- A guided proposal, using the latest observation to steer particles toward regions of high posterior probability.
- A tempering procedure to help bridge the gap between the proposal and the target distribution.

- An MCMC mutation step to jitter samples and ensure particle diversity.

We introduce each of these ideas below and bring them together to form a new method for the filtering of discretely observed SDEs. We refer to this combined methodology as the Guided Proposal with Tempering and Mutation Filter (GPTMF)

3.1 The proposal step

The standard bootstrap filter proposes samples from the prior distribution defined by the underlying SDE (1.63) and is blind to the observed data point, Y_k. The proposals can clearly be improved by taking the new data point into account and guiding particles toward the data and into regions of higher probability given the observed data.

At each time we wish to generate proposals that approximate the law of our underlying SDE (1.63), of the form

$$dX_t = b\left(X_t\right) + \sigma\left(X_t\right) dW_t,$$

over the time interval $[0, T]$, conditioned on $X_0 = x_0 \in \mathbb{R}^d$ and an observation at time T, y_T, observed with Gaussian noise as defined in (1.64).

Following the methods of Schauer et al. [2017] we can build such a proposal as

$$dZ_t = b(Z_t)dt + \sigma(Z_t)\sigma(Z_t)^T \nabla_x \log \tilde{p}(y_T, T|Z_t, t)dt + \sigma(Z_t)dW_t, \tag{3.1}$$

where $\tilde{p}(y, T|z, t)$ is the likelihood of observing the data point given that the underlying SDE is a Brownian motion, i.e. $b = 0, \sigma = I_d$. It is straightforward to show [Golightly and Wilkinson, 2008] that this likelihood corresponds to the density of the Gaussian, $N(z, \Sigma + (T-t)I_d)$, where Σ is the covariance matrix of the noise in the observations. From here we get that

$$\nabla_x \log \tilde{p}(y, T|z, t) = (\Sigma + (T-t)I_d)^{-1}(y - z) \tag{3.2}$$

and (3.1) becomes

$$dZ_t = \tilde{b}(Z_t, t, y_T, T)dt + \sigma(Z_t)dW_t \tag{3.3}$$

where

$$\tilde{b}(z, t, y, T) := b(z) + \sigma(z)\sigma(z)^T(\Sigma + (T-t)I_d)^{-1}(y - z). \tag{3.4}$$

The use of (3.3) to generate proposals is justified by Schauer et al. [2017], in the sense that the law of the process (3.3) is absolutely continuous with respect to the law of the target process (1.63) and an expression for the Radon-Nikodym derivative can be found.

3.1.1 Euler-Maruyama method

In order to sample from the proposal we are required to numerically approximate the solution of a given SDE. There are numerous ways to do this (see Kloeden and Platen [2013]) but for our purposes we restrict ourselves to the Euler-Maruyama scheme, which is fast and easy to implement.

Suppose we have an SDE of the form (1.63) with initial condition $X_0 = x_0$ and we would like to approximate a solution on the time interval $[0, T]$. We partition the time interval into N sections of equal length $0 = t_0 < \cdots < t_N = T$ and recursively define

$$X_{t_{i+1}} = X_{t_i} + b(X_{t_i})(t_{i+1} - t_i) + \sigma(X_{t_i})(W_{t_{i+1}} - W_{t_i}), \tag{3.5}$$

where W_t is a standard Brownian motion. This gives a discretised approximation to a solution of (1.63) with weak approximation error that is $\mathcal{O}(\Delta t)$, and strong approximation error that is $\mathcal{O}(\sqrt{\Delta t})$. These convergence rates hold under standard regularity conditions, namely that both the drift and diffusion coefficients satisfy both a global Lipschitz condition and a linear growth bound. [Kloeden and Platen, 2013].

3.2 The tempering step

Denote by $\mathbb{X}_{x,j}$ the distribution on the space of continuous paths $C[t_{j-1}, t_j]$ induced by (1.63) for initial value $X_{t_{j-1}} = x$, define $\mathbb{Z}_{x,j}$ similarly for the distribution induced by (3.3).

At the jth time step of a particle filter such as Algorithm (2.3) the target distribution of the sampler is $\mathbb{X}_{x,j}(dX|y_j)$ - the distribution $\mathbb{X}_{x,j}(dX)$, conditioned on the observation y_j. In the straightforward setup of Algorithm (2.3) the idea is to use proposals from $\mathbb{Z}_{x,j}$ as candidate paths from the target and weight these paths according to the Radon-Nikodym derivative

$$\frac{d\mathbb{X}_{x,j}(X|y_j)}{d\mathbb{Z}_{x,j}(X)} = G_{x,j}(X, y_j), \tag{3.6}$$

where

$$G_{x,j}(X, y_j) = \frac{d\mathbb{X}_{x,j}(X)}{d\mathbb{Z}_{x,j}(X)} p(y_j \mid X_{t_j}). \tag{3.7}$$

We then have a set of weighted samples as an approximation to the target distribution.

Issues can arise with this method if the proposal is not similar enough to the target distribution and the particle weights can soon degenerate and become dominated by a relatively small number of particles. To try and combat this we can supplement Algorithm (2.3) with a tempering procedure - the idea is to use a sequence of temperatures

$$0 = \phi_0 < \phi_1 < \cdots < \phi_K = 1 \tag{3.8}$$

which give rise to a sequence of intermediate path distributions

$$\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_0}, \mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_1}, \ldots, \mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_K}, \tag{3.9}$$

note that $\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_0} = \mathbb{Z}_{x,j}(dX)$ and $\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_K} = \mathbb{X}_{x,j}(dX|y_j)$.

These distributions act as a bridge between our proposal and target distributions, and we can perform iterative importance sampling to reach the target. That is, at each step of the tempering procedure we perform importance sampling with proposal distribution $\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_{l-1}}$ and target distribution $\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_l}$, with the idea being that these distributions are similar enough to preserve some level of stability in the particle weights.

The temperatures ϕ_l are determined on the fly and chosen to be as large as possible while preserving a minimum ESS at each step of the procedure (e.g. via bisection search).

3.3 The mutation step

3.3.1 MCMC on general spaces

Markov Chain Monte Carlo (MCMC) methods were developed to generate samples from target distribution with density π that cannot be sampled using standard methods. The approach constructs a reversible Markov chain with π as its stationary distribution. By running this chain until equilibrium and collecting its trajectory, we obtain correlated samples from π.

Perhaps the most commonly used method of constructing a Markov chain with a specified target distribution is the standard Metropolis-Hastings [Hastings, 1970, Metropolis et al., 1953] algorithm. For this we require a transition kernel $q(dx'|x)$, from which we draw a candidate for the next state, this candidate is then accepted with probability

$$\alpha(x, x') = 1 \wedge \frac{\pi(x')q(x'|x)}{\pi(x)q(x|x')}. \tag{3.10}$$

This process is repeated until we are satisfied that the chain has converged to its stationary distribution, we then continue the process to draw samples from π.

In the above presentation it is assumed that all densities present in (3.10) are all with respect to a common reference measure. For state spaces of finite dimension this is typically the Lebesgue measure on $\mathbb{R}^d$. In more complex state spaces it is more typical that a common reference measure cannot be found. For instance, the state space relevant to our use case is $C[0,t]$, the infinite-dimensional space of continuous paths on $\mathbb{R}^d$ over some interval $[0,t]$. For Markov kernels on this space it is often the case that the proposal transitions are singular for different current states, i.e. the probability measures $q(dx'|x_1)$ and $q(dx'|x_2)$ are mutually singular for $x_1 \neq x_2$.

For the construction of Metropolis-Hastings algorithms in these more complex state spaces we refer to the more general theory developed in Tierney [1998]. For a target distribution with density $\pi(dx)$ and transition kernel $q(dx'|x)$ define the bivariate distributions

$$\mu(dx, dx') := \pi(dx)q(dx'|x), \tag{3.11}$$

$$\mu^T(dx, dx') := \pi(dx')q(dx|x'). \tag{3.12}$$

From Tierney [1998] we have that if μ and μ^T are absolutely continuous with respect to each other then the acceptance probability, $\alpha(x, x')$, is well specified, in the sense that it is μ-a.s. positive (as opposed to the case where μ and μ^T are mutually singular and the acceptance probability is μ-a.s. zero.) and we have

$$\alpha(x, x') = 1 \wedge \frac{d\mu^T}{d\mu}(x, x'). \tag{3.13}$$

Therefore, calculating acceptance probabilities for MCMC kernels in complex state spaces requires finding the Radon-Nikodym derivative $\frac{d\mu^T}{d\mu}$ in (3.13).

3.3.2 MCMC on pathspace

We return now to the task of finding a Markov kernel to mutate the particles in our tempering procedure. We wish to specify kernels $K_{j,l}(dX', X)$, that preserve the distribution $\mathbb{Z}_{x,j}(dX)G_{x,j}(X, y_j)^{\phi_l}$, denote this distribution by $\mathbb{P}^l_j$ and note that it is defined such that

$$\frac{d\mathbb{P}^l_j}{d\mathbb{Z}_{x,j}}(X) \propto G_{x,j}(X, y_j)^{\phi_l}, \tag{3.14}$$

where $\mathbb{Z}_{x,j}$ corresponds to the law of the process

$$dZ_t = \tilde{b}(Z_t, t, y_{t_j}, t_j)dt + \sigma(Z_t)dW_t, \quad t \in [t_{j-1}, t_j], \quad Z_{t_{j-1}} = x \tag{3.15}$$

Under the assumption that σ is invertible and other regulatory conditions which we omit for brevity (3.15) gives rise to a 1-1 mapping [Golightly and Wilkinson, 2008]:

$$W \to F(W, x, y_j) := Z \tag{3.16}$$

The paths generated by solutions of (3.15) are uniquely determined by the realisations of the Brownian motion W. Denote by $\mathbb{W}$ the law of the standard Brownian motion on $\mathbb{R}^d$ and we immediately have $\mathbb{Z}_{x,j} \circ F \equiv \mathbb{W}$. Let $\tilde{\mathbb{P}}^l_j$ denote the law of $F^{-1}(X)$ for $X \sim \mathbb{P}^l_j$, then by the absolute continuity of $\mathbb{P}^l_j$ with respect to $\mathbb{Z}_{x,j}$ and the preservation of measure under one-to-one transformations we have

$$\frac{d\tilde{\mathbb{P}}_j^l}{d\mathbb{W}}(W) \equiv \frac{d\mathbb{P}_j^l}{d\mathbb{Z}_{x,j}}(F(W,x,y_j)) \propto G_{x,j}(F(W,x,y_j),y_j)^{\phi_l}. \tag{3.17}$$

The preservation of $\mathbb{P}_j^l$ is equivalent to preserving $\tilde{\mathbb{P}}_j^l$ under the transformation, F, and we can mutate our paths by mutating the driving Brownian motion.

In (3.17) we have expressed $\tilde{\mathbb{P}}_j^l$ as a change of a measure from a Gaussian distribution. Much work has been done in developing well specified MCMC kernels on infinite-dimensional pathspaces for targets in this family of distributions, including Metropolis-adjusted Langevin algorithm (MALA) [Roberts et al., 1996] and Hybrid Monte Carlo (HMC) methods [Duane et al., 1987, Beskos et al., 2013]. For the purposes of this thesis we use the Preconditioned Crank-Nicolson (pCN) method [Beskos et al., 2008b, Cotter et al., 2013].

We wish to provide an MCMC kernel that preserves $\tilde{\mathbb{P}}_j^l$ and we have shown that it can be expressed as a change of measure from a Gaussian law. For target distribution $\tilde{\mathbb{P}}_j^l$ and proposal kernel $Q(dW'|W)$ specified by the proposal

$$W' = \rho W + \sqrt{1-\rho^2}\xi, \quad \xi \sim N(0,\mathcal{C}) \tag{3.18}$$

where $\mathcal{C}$ is the covariance of the Brownian motion and ρ is a parameter that controls the step size, it follows from Cotter et al. [2013] Theorem 6.2 that the measures $\eta(dW,dW') := \tilde{\mathbb{P}}_j^l(dW)Q(dW'|W)$ and $\eta^T(dW,dW') := \tilde{\mathbb{P}}_j^l(dW')Q(dW|W')$ are mutually absolutely continuous, with Radon-Nikodym derivative

$$\frac{d\eta^T}{d\eta}(W, W') = \frac{G_{x,j}(Z', y_j)^{\phi_l}}{G_{x,j}(Z, y_j)^{\phi_l}}, \qquad Z = F(W, x, y_j), \qquad Z' = F(W', x, y_j). \tag{3.19}$$

We now have an MCMC transition kernel that preserves $\tilde{\mathbb{P}}^l_j$ (and hence preserves $\mathbb{P}^l_j$ via the F-transform), defined via proposals as in (3.18) and acceptance probability

$$\alpha(W, W') = 1 \wedge \frac{G_{x,j}(Z', y_j)^{\phi_l}}{G_{x,j}(Z, y_j)^{\phi_l}}. \tag{3.20}$$

3.4 The algorithm

We are now ready to put the above methods together and describe a particle filtering algorithm with tempering and mutation steps for filtering of partially observed SDEs. The algorithm is outlined in Algorithm 3.1.

Algorithm 3.1: Particle filter with tempering

```
1. For $i = 1, \dots, N$, sample $x_0^i \sim p(x_0)$. Set $j = 1$.
2. For current particles $\{x_{j-1}^i\}_{i=1}^N$, set $\phi_0 = 0, l = 1$.
      (i) For $i = 1, \dots, N$ generate sample paths on $[t_{j-1}, t_j]$ as
          $X^i \sim \mathbb{Z}_{x_{j-1}^i, j}$.
     (ii) For $\phi > \phi_{l-1}$ determine weights
              $$w_l^i(\phi) = G_{x_j^i, j}(X^i, y_j)^{\phi - \phi_{l-1}}$$
    (iii) Find $\phi > \phi_{l-1}$ such that $ESS(\{w_l^i(\phi)\}_{i=1}^N) = ESS_{\text{min}}$ for some
          predetermined $ESS_{\text{min}}$. Set $\phi_l = 1 \wedge \phi$.
     (iv) Resample $\{X^i\}_{i=1}^N$ according to $\{w_l^i(\phi)\}_{i=1}^N$. Set $\{\bar{X}^i\}_{i=1}^N$ to be
          the set of resampled paths.
      (v) For $i = 1, \dots, N$, mutate $X^i \sim K_{j,l}(dX', \tilde{X}^i)$ for kernel
          specified by pCN scheme to preserve $\mathbb{Z}_{x_{j-1}^i, j}(dx) G_{x_j^i, j}(x, y_j)^{\phi_l}$.
     (vi) If $\phi < 1$, set $l = l + 1$ and return to step 2(ii); else proceed.
3. Output particles $\{x_j^i, w_j^i\}_{i=1}^N$, where $x_j^i = X_{t_j}^i$.
4. Set $j = j + 1$, if $j > n$ end, else return to step 2.
```

Chapter 4

Numerical experiments on double-well potential

We compare the results of the GPTMF in Algorithm (3.1) with results obtained by applying the standard bootstrap algorithm.

We run both filters from $t = 0$ to $t = 10$, with a time between observations of 0.1.

In all cases we run the algorithms with 1000 particles, resampling in the bootstrap algorithm is triggered when the ESS falls below 500 and temperatures are found in the tempering step such that the ESS stays above the same value. The value of ρ for pCN steps (as in (3.18)) is 0.99 in all cases and we carry out five pCN steps to generate a single mutation step.

We consider a d-dimensional process where each dimension is an independent double-well process. That is the filtering problem with underlying SDE

$$dX_t = b(X_t)dt + dW_t, \tag{4.1}$$

where

$$b(X_{t,1}, \cdots, X_{t,d}) = \begin{bmatrix} 4X_{t,1}(1 - X_{t,1}^2) \\ \vdots \\ 4X_{t,d}(1 - X_{t,d}^2) \end{bmatrix}, \tag{4.2}$$

and we have observations

$$Y_k = X_{t_k} + N(0, \Sigma), \tag{4.3}$$

with $\Sigma = 0.01 I_d$.

Figure 4.1 shows point estimates and confidence intervals for the first coordinate of a double-well process, calculated for problems of different dimensions using both the bootstrap and GPTMF. It is clear to see the deterioration in performance of the bootstrap filter as we increase the dimension of the problem. This deterioration shows in two ways: first, the point estimates themselves fail to accurately track the true state. Second, the confidence intervals shrink as the weights are dominated by just one or two particles. We can see, particularly in the $d = 10$ case, that many of the actual states of the system are outside these artificially narrow confidence intervals.

GPTMF on the other hand appears to be much more robust to the dimension of the problem. The point estimates maintain good tracking accuracy even in high dimensions. Additionally, the confidence intervals maintain a reasonable width that properly reflects the posterior uncertainty, with the majority of the actual states of the system found inside these confidence intervals. Figure 4.2 shows this difference very clearly, as we increase the dimension of the SDE the proportion of actual states of the system at the observation times that can be found inside the confidence intervals generated from the bootstrap methodology decreases dramatically, while for the method that makes use of tempering and mutation this proportion stays relatively stable.

Figure 4.3 shows how, as expected, the number of temperatures required in the tempering step increases with the dimension of the SDE. As the dimensionality increases we expect to see the weights in a particle filter dominated by fewer particles, so it makes sense that as we increase the dimension we would have to increase the number of temperatures required to bridge the gap between our proposal and posterior distributions.

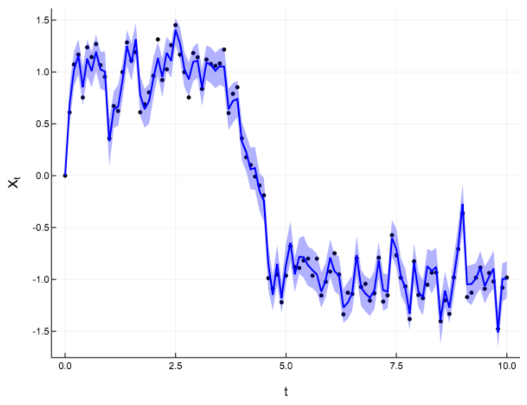
Each tempering step adds a computational load, involving weight calculations, resampling, and MCMC mutations for all N particles. Therefore, the total cost of the GPTMF per time step is significantly higher than that of the bootstrap filter and grows with the dimension. However, the payoff is that the GPTMF yields accurate and reliable estimates in settings where the bootstrap filter fails completely.

While we do not have a precise scaling law for how the number of temperatures scales with the dimension of the SDE, Figure 4.3 suggest a growth that is sublinear. This indicates that while the computational cost increases with dimensionality, it may do so at a manageable rate, making the approach feasible for high-dimensional problems.

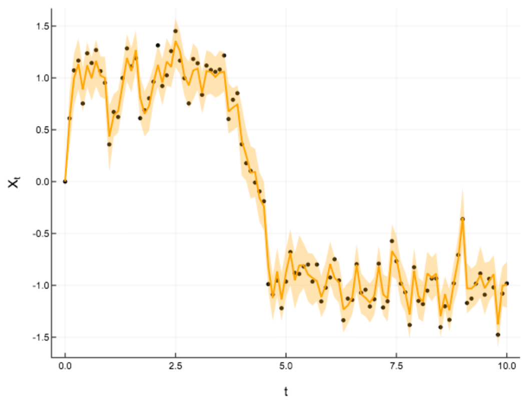
Figure 4.4 shows the value of the first coordinate of our proposal paths pre and post mutation at various times. It is clear to see there is a strong positive, but not perfect correlation between the values. This corresponds to the kind of mixing we expect to see from our pCN kernel. We do not want to move the particles significantly, just jitter them slightly to increase the diversity in the particle set.

Figure 4.5 shows that mixing effectiveness decreases with dimension. This suggests that higher dimensional problems may benefit from additional pCN steps to achieve comparable mixing quality, though at increased computational cost. Future work could explore adaptive schemes that adjust the number of mutation steps based on dimension or monitored mixing diagnostics.

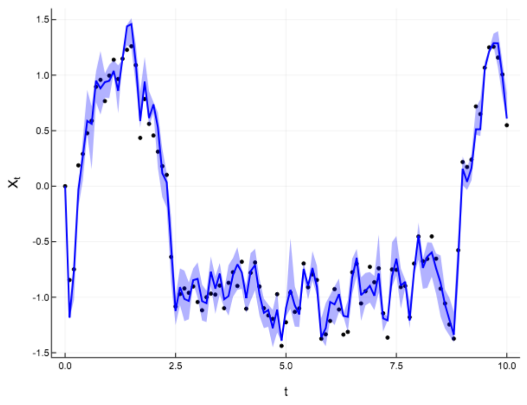
The nature of the double-well problem (4.2), means that each coordinate is independent of all others. Because of this we could just run a separate, one-dimensional particle filter, for each coordinate. This allows us to run a bootstrap particle filter with a large number of particles to establish a highly accurate benchmark for the posterior distribution for a given coordinate. To test the effectiveness of our tempering algorithm we run a bootstrap filter (effective for a one-dimensional problem) with 10000 particles on the first coordinate of our problem and compare with the estimated posterior for that coordinate from our tempering algorithm run on the ten-dimensional problem with just 1000 particles, the results of this can be seen in figure 4.6. It's clear to see that the tempering algorithm produces reasonable estimates of the posterior when compared to this benchmark posterior.



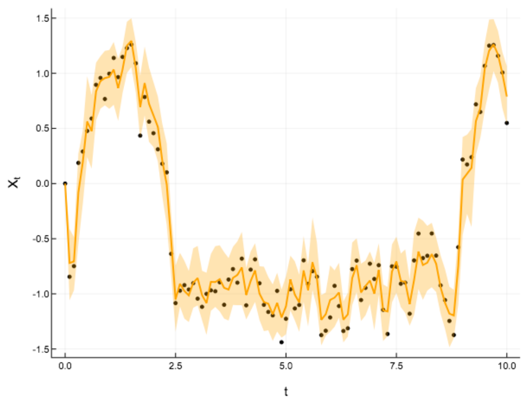
(a) $d = 2$, bootstrap



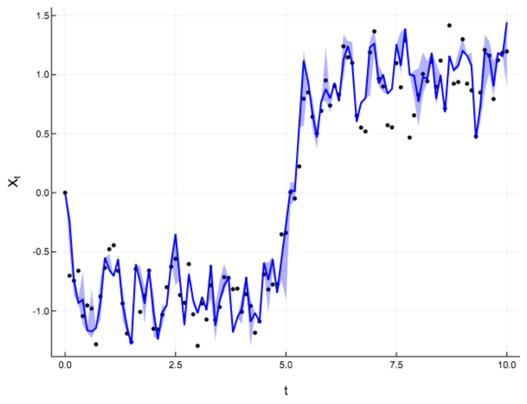
(b) $d = 2$, GPTMF



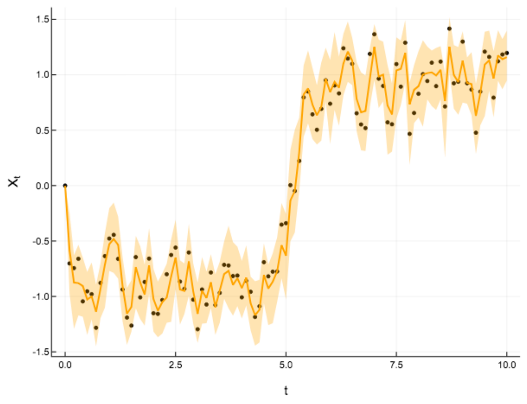
(c) $d = 6$, bootstrap



(d) $d = 6$, GPTMF



(e) $d = 10$, bootstrap



(f) $d = 10$, GPTMF

Figure 4.1: Estimates and confidence intervals of the first coordinate of double-well processes at different dimensions. The black dots are the true states of the hidden process X_t at observation times, the coloured line is the point estimate of the state of the actual process and the shaded area correspond to 95% confidence intervals.

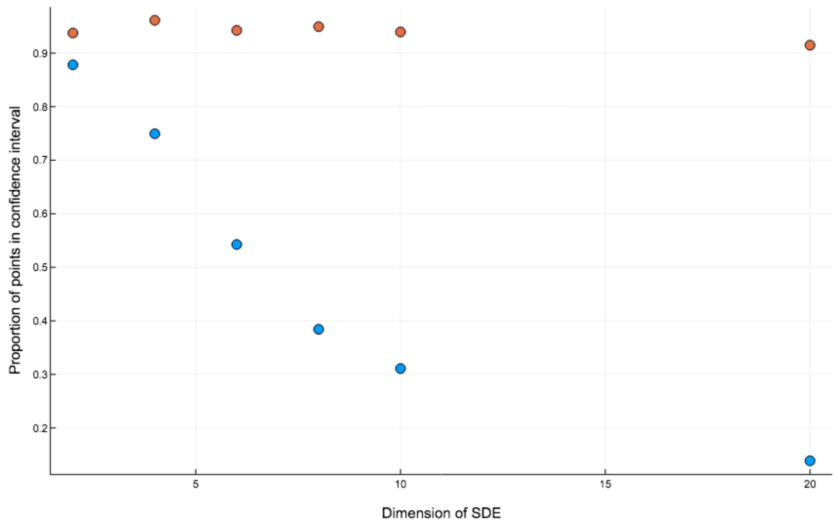


Figure 4.2: Above shows the proportion of 95% confidence intervals that contain the actual state of the system at the observation times, for confidence intervals generated using the bootstrap (blue) estimates and the GPTMF (orange) estimates. These values were calculated using ten independent runs of the algorithm for each of the dimensions 2,4,6,8,10 and 20.

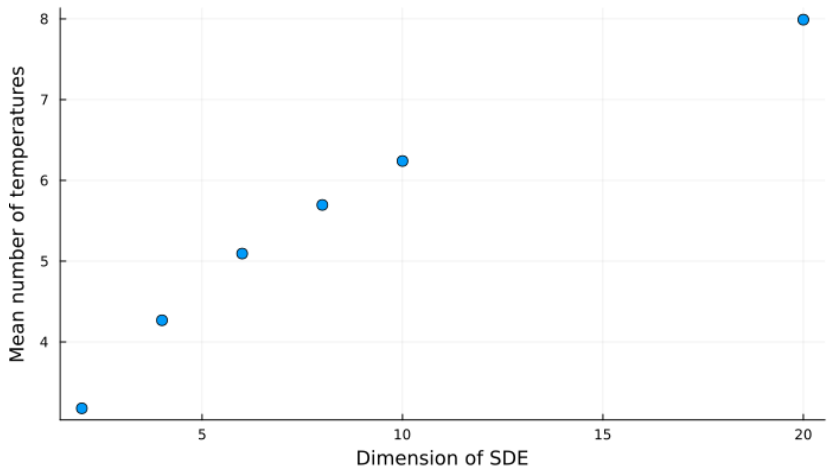
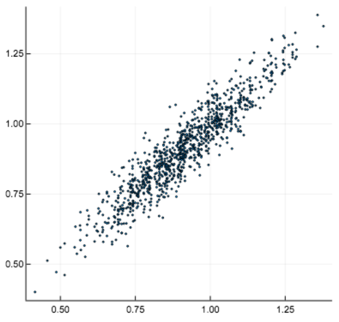
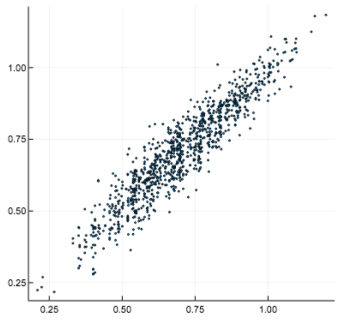
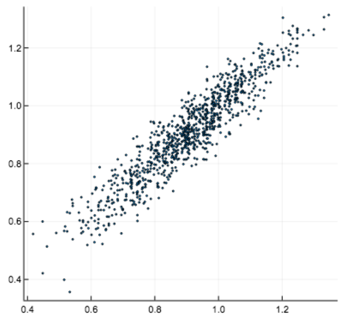
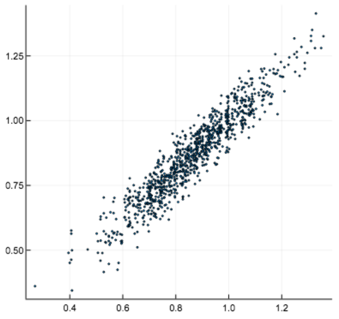


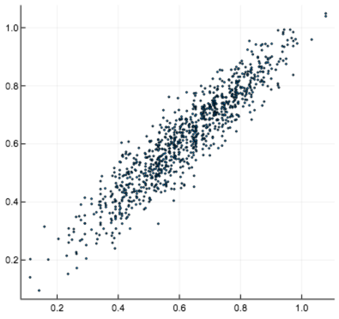
Figure 4.3: The above shows the average number of temperatures per time step against the dimension of the SDE. These values were calculated using ten independent runs of the algorithm for each of the dimensions 2,4,6,8,10 and 20.

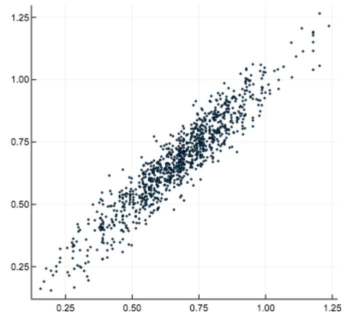

(a) $t = 2.0$

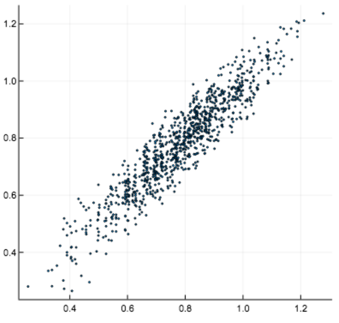

(b) $t = 3.0$

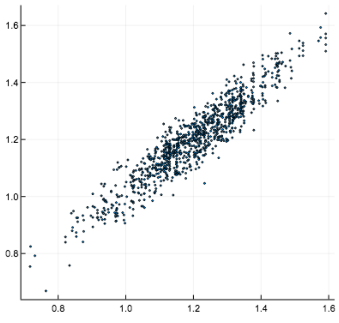

(c) $t = 4.0$


(d) $t = 5.0$


(e) $t = 6.0$


(f) $t = 7.0$


(g) $t = 8.0$


(h) $t = 9.0$

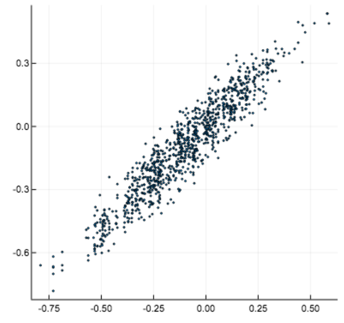

(i) $t = 10.0$

Figure 4.4: Above shows the positions of the endpoint of the first coordinate of proposal paths in the GPTMF algorithm, pre-mutation is on the x-axis and post mutation is on the y-axis. These results were taken from a single run of the PF algorithm on a six dimensional double-well problem.

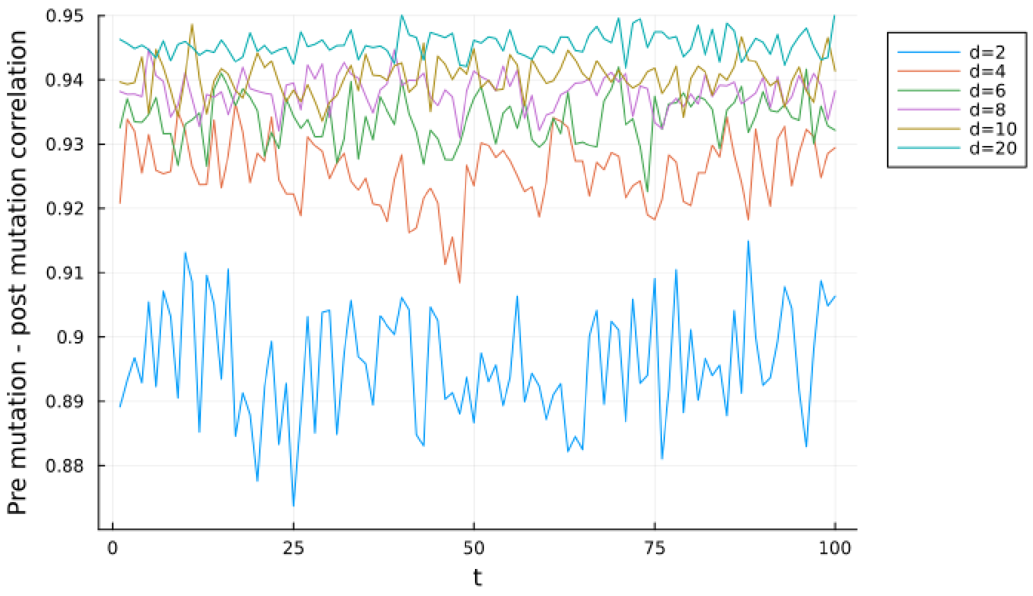
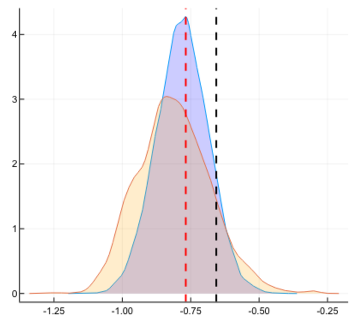
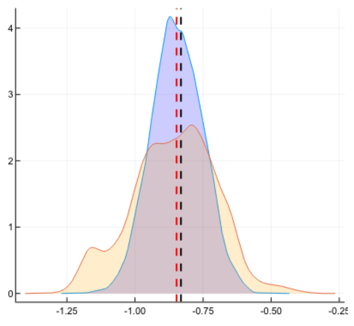
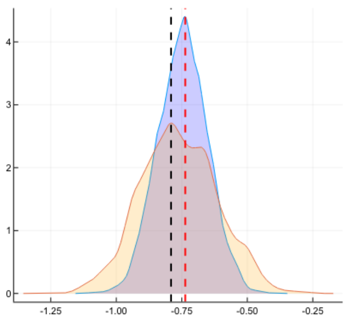
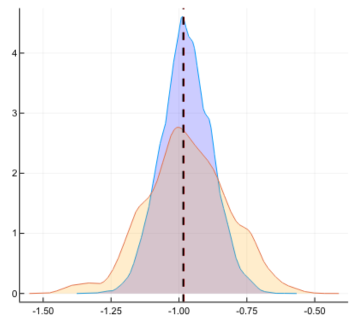


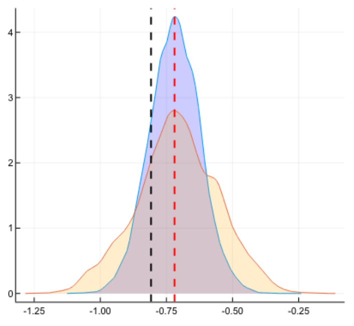
Figure 4.5: Pearson's correlation coefficient of the pre and post mutation positions of the first coordinate of the particles at each observation time. These results were taken from a single run of the GPTMF algorithm for dimensions 2,4,6,8,10 and 20.

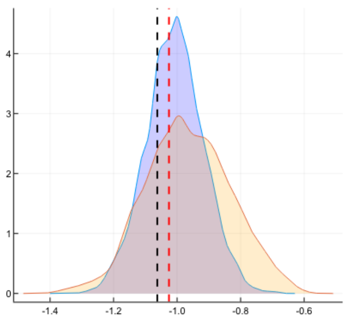

(a) $t = 2.0$

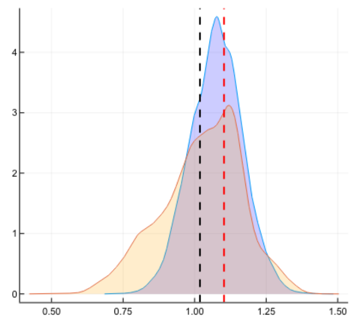

(b) $t = 3.0$

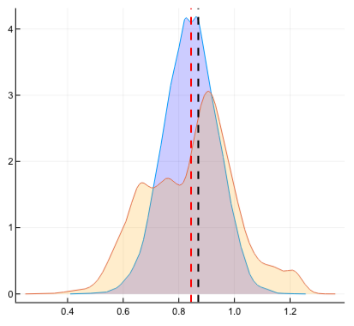

(c) $t = 4.0$


(d) $t = 5.0$


(e) $t = 6.0$


(f) $t = 7.0$


(g) $t = 8.0$


(h) $t = 9.0$

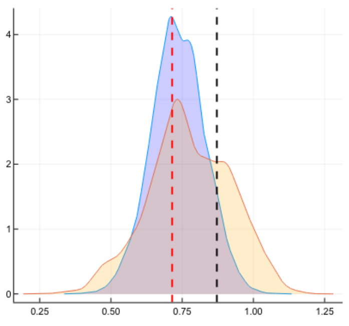

(i) $t = 10.0$

Figure 4.6: Estimated densities of the posterior from the one-dimensional bootstrap algorithm with 10000 particles (blue) compared with the estimated posterior from the GPTMF run on the ten-dimensional problem with 1000 particles (orange). The black dashed line shows the true value of the underlying system at that time and the red dashed line shows the observed value of the system.

Chapter 5

Data Assimilation of Tsunami Wavefields

Early detection of tsunamis can help to enable timely reactions to the situation, such as evacuating coastal areas, that help to minimise the loss of life and damage to property. Key to being able to accurately forecast the incidence of a wave at a coastline is the ability to assimilate data into the models in real time.

Many detection schemes require the use of data from seismic events, this may lead them to overlook tsunamis caused by other types of event (e.g. those caused by landslides or volcanic eruptions). We follow the work of Maeda et al. [2015] and present a scheme that uses a network of tsunameters measuring wave heights to estimate the height and velocity of the tsunami. In contrast to Maeda et al. [2015], who assimilate data via a Kalman filter, we present a particle filter based approach.

5.1 Tsunami PDE model

As in Maeda et al. [2015] the behaviour of the waves is assumed to be governed by a simple 2-D linear long-wave equation

$$\begin{aligned}\frac{\partial \eta(x,y,t)}{\partial t} &= -\frac{\partial M(x,y,t)}{\partial x} - \frac{\partial N(x,y,t)}{\partial y}\\ \frac{\partial M(x,y,t)}{\partial x} &= -gD(x,y)\frac{\partial \eta(x,y,t)}{\partial x}\\ \frac{\partial N(x,y,t)}{\partial y} &= -gD(x,y)\frac{\partial \eta(x,y,t)}{\partial y},\end{aligned} \tag{5.1}$$

where η is the tsunami height, M, N are the vertical and horizontal components of the tsunami velocity, g is the gravitational constant and $D(x,y)$ is the depth of the ocean.

5.1.1 Boundary conditions

Suppose we simulate the above equations on a grid, the dynamics we are interested in should occur far from the boundary of the grid and we seek boundary conditions such that the dynamics far from the edge of the grid are not significantly affected. To achieve this we adopt the non-reflecting boundary condition from Cerjan et al. [1985], outlined below.

Suppose we simulate on a 2D domain $\Omega = [0,P] \times [0,Q]$, where we discretise Ω in space to form a grid of $(p+1)\times(q+1)$ equally spaced points, i.e. the grid

$$\Omega_g := \{(w_{1,i}, w_{2,j}) : w_{1,i} = i\Delta_p, w_{2,j} = j\Delta_q, 0 \leqslant i \leqslant p, 0 \leqslant j \leqslant q\}, \tag{5.2}$$

where $\Delta_p = P/p, \Delta_q = Q/q$.

The boundary conditions are then applied via the matrix, B, where for a point $(w_{1,i}, w_{2,j})$ in Ω_g the corresponding matrix entry is defined as:

$$B(i,j) = \exp\left[-(\lambda \max\{0, d_s - d_e(i,j)\})^2\right] \tag{5.3}$$

where

1. λ is a damping factor, decided by the user.
2. d_s is the size of the boundary, i.e. the number of points from the edge of the grid that boundary damping will begin.
3. $d_e(i,j)$ is the number of points to the closest edge of the grid (vertically or horizontally), i.e.

$$d_e(i,j) = \min(i, p-i, j, q-j).$$

Essentially the wave begins to be damped as it gets within a specified distance of the edge of the grid and this damping gets exponentially stronger as you approach the edge of the grid.

5.1.2 Simulation

We simulate the system of equations 5.1 via a simple finite difference scheme, outlined below.

Suppose that we are working on a grid Ω_g as defined above (5.2), and suppose that we know the height and velocity fields, η, M, N at time t. We evolve the system to time $t+1$ as follows:

Define

$$\Delta_x \eta_t(w_{1,i}, w_{2,j}) := \frac{\eta_t(w_{1,i}, w_{2,j}) - \eta_t(w_{1,i-1}, w_{2,j})}{\Delta_p}$$

and similarly

$$\Delta_y \eta_t(w_{1,i}, w_{2,j}) := \frac{\eta_t(w_{1,i}, w_{2,j}) - \eta_t(w_{1,i}, w_{2,j-1})}{\Delta_q}.$$

Update velocities:

$$\begin{aligned} M_{t+1}(w_{1,i}, w_{2,j}) &= B(i,j)\Big[M_t(w_{1,i}, w_{2,j}) - gD(w_{1,i}, w_{2,j})\Delta_x \eta_t(w_{1,i}, w_{2,j})\Delta t\Big] \\ N_{t+1}(w_{1,i}, w_{2,j}) &= B(i,j)\Big[N_t(w_{1,i}, w_{2,j}) - gD(w_{1,i}, w_{2,j})\Delta_y \eta_t(w_{1,i}, w_{2,j})\Delta t\Big]. \end{aligned}$$

Then define

$$\begin{aligned} \Delta M_{t+1}(w_{1,i}, w_{2,j}) &= \frac{M_{t+1}(w_{1,i+1}, w_{2,j}) - M_{t+1}(w_{1,i}, w_{2,j})}{\Delta_p} \\ \Delta N_{t+1}(w_{1,i}, w_{2,j}) &= \frac{N_{t+1}(w_{1,i}, w_{2,j+1}) - N_{t+1}(w_{1,i}, w_{2,j})}{\Delta_q}. \end{aligned}$$

Finally we can update the height as:

$$\eta_{t+1}(w_{1,i}, w_{2,j}) = B(i,j)\left[\eta_t(w_{1,i}, w_{2,j}) - \Delta_t\Big(M_{t+1}(w_{1,i}, w_{2,j}) + N_{t+1}(w_{1,i}, w_{2,j})\Big)\right].$$

5.1.3 Initialisation of tsunami

To simulate a tsunami we require initial conditions that artificially simulate the kind of event that would usually cause a tsunami. Following Maeda et al. [2015], we achieve this by starting with our initial sea height extremely high in one area of our domain. The water in this area then falls into the sea, creating a wave that propagates across our domain.

More precisely, if we want to initiate a wave centred at a point $(x_c, y_c), 0 \leq x_c \leq P, 0 \leq y_c \leq Q$, with initial spread s and initial maximum height h_{max} then the initial height field can be calculated as

$$\eta_0(x, y) = \begin{cases} h_{\text{max}} h_{x_c,s}(x) h_{y_c,s}(y) & \text{if } |x - x_c| \leq s \text{ and } |y - y_c| \leq s \\ 0 & \text{otherwise} \end{cases}, \tag{5.4}$$

where

$$h_{a,s}(z) := \frac{1 + \cos(\pi \frac{z-a}{s})}{2}. \tag{5.5}$$

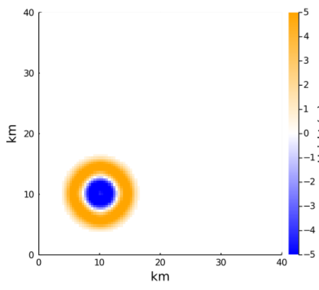
Initial values for velocities in both the x and y directions are set to 0.

5.2 Discrete time model noise

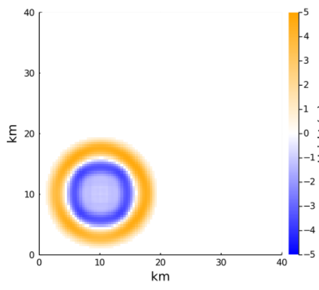
5.2.1 Model dynamics

We first consider the case of adding model noise to the system (5.1) via the addition of noise at discrete time steps. Between these time steps we solve the system deterministically. This gives us the dynamics:

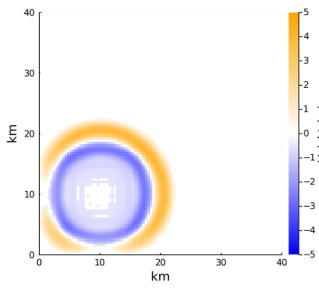
$$\begin{aligned} x_0 &\sim p(dx_0) \\ x_t &= F(x_{t-1}) + \xi_t \\ y_t &= Ax_t + \zeta_t, \end{aligned} \tag{5.6}$$



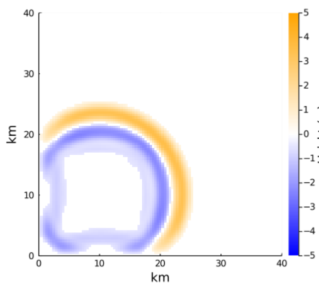
(a) $t = 25s$



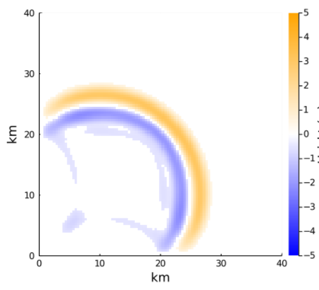
(b) $t = 50s$



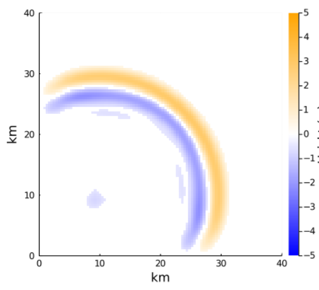
(c) $t = 75s$



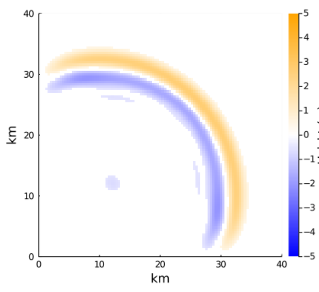
(d) $t = 100s$



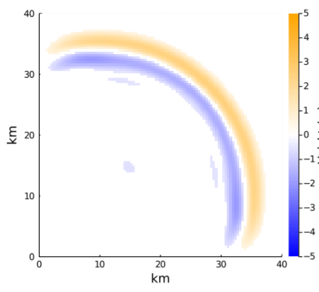
(e) $t = 125s$



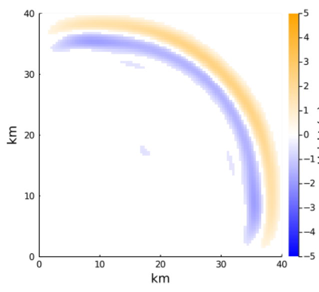
(f) $t = 150s$



(g) $t = 175s$



(h) $t = 200s$



(i) $t = 225$

Figure 5.1: Example of a deterministic tsunami simulation driven by the 2-D linear long wave equation(5.1) *. Simulated on a regular 100 $\times$ 100 grid and with a time step of 0.1 seconds.*

for a mapping $F : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, linear operator $A \in \mathbb{R}^{d_y \times d_x}$ and iid sequences of noise

$$\xi_t \sim N(0, \Sigma)$$
$$\zeta_t \sim N(0, S)$$

for covariance matrices $\Sigma \in \mathbb{R}^{d_x \times d_x}, S \in \mathbb{R}^{d_y \times d_y}$.

In this particular context of data assimilation for a tsunami wavefield, the mapping F corresponds to moving the system forward via the dynamics in (5.1), the linear operator A selects the locations at which we have observation stations and the model noise ξ is sampled from some stationary Gaussian field with covariance function $r(x, y)$. We aim to construct a particle filter for online estimation of the filtering distribution, $p(dx_t|y_{1:t})$.

Although we could apply the bootstrap filter to estimate the filtering distribution, we anticipate poor performance due to the high dimensionality of the tsunami wavefield.

5.2.2 Optimal filter

The particular structure of the dynamics (5.6) allows us to use what is known in the literature as the optimal filter [Doucet et al., 2000], specifically a particle filter with proposal distribution

$$q(x_t|x_{t-1}, y_t) := p(x_t|x_{t-1}, y_t) \tag{5.7}$$

and corresponding weight update

$$\begin{aligned} w_t &= w_{t-1}\frac{p(x_t|x_{t-1})p(y_t|x_t)}{p(x_t|x_{t-1},y_t)} \\ &= w_{t-1}p(y_t|x_{t-1}). \end{aligned} \tag{5.8}$$

Since the observation operator A is linear and both noise terms are additive Gaussian, the distributions $p(x_t|x_{t-1},y_t)$ and $p(y_t|x_{t-1})$ are analytically tractable. Direct calculation gives

$$p(dy_t|x_{t-1}) = N\left(AF(x_{t-1}), A\Sigma A^T + S\right) \tag{5.9}$$

and

$$p(dx_t|x_{t-1},y_t) = N\left(V(\Sigma^{-1}F(x_{t-1}) + A^TS^{-1}y_t), V\right), \tag{5.10}$$

where $V := (\Sigma^{-1} + A^TS^{-1}A)^{-1}$.

5.2.3 Computational setup

As before, we assume we are considering the dynamics on a 2D-domain, $\Omega = [0,P] \times [0,Q]$ and that the domain is discretised to form a grid Ω_g, as in 5.2, of size $n_g = (p+1) \times (q+1)$.

The state of the system at time t is represented by a vector $x_t \in \mathbb{R}^{3n_g}$, where we have mapped the three-dimensional field on Ω_g representing wave height, horizontal velocity and vertical velocity to a single vector of length $3n_g$. The first n_g entries correspond to the wave heights, the second n_g entries correspond to the horizontal velocity and the final n_g entries correspond to the vertical velocity.

With the above correspondence, we consider the mapping $F : \mathbb{R}^{3n_g} \to \mathbb{R}^{3n_g}$ (in our case this is solving equations 5.1 forward in time) via its effect on the three fields. That is, we rewrite F as

$$F(x_t) = \begin{bmatrix} F_1(x_t) \\ F_2(x_t) \\ F_3(x_t) \end{bmatrix}. \tag{5.11}$$

We assume our d_y observation stations are located at fixed points located on the grid Ω_g, producing a vector, $y_t \in \mathbb{R}^{d_y}$, of height observations. The linear operator A selects these points from our grid. As we have no observations of velocity, A can be represented by a $d_y \times 3n_g$ matrix of the form

$$A = \begin{bmatrix} A_1 & 0_{d_y \times n_g} & 0_{d_y \times n_g}, \end{bmatrix} \tag{5.12}$$

where A_1 is a $d_y \times n_g$ matrix with zeros everywhere, except for a single entry of 1 on each row, each corresponding to the position of an observation station.

We assume the observation errors are independent across stations and that each follows a Gaussian distribution with mean 0 and variance c^2, so we can write the covariance matrix, S, as $S = c^2 I_{d_y}$. We also assume that the model noise is independent across the three height and velocity fields, and we can write the covariance matrix as

$$\Sigma = \begin{bmatrix} \Sigma_1 & 0_{n_g \times n_g} & 0_{n_g \times n_g} \\ 0_{n_g \times n_g} & \Sigma_2 & 0_{n_g \times n_g} \\ 0_{n_g \times n_g} & 0_{n_g \times n_g} & \Sigma_3, \end{bmatrix} \tag{5.13}$$

where $\Sigma_1, \Sigma_2, \Sigma_3 \in \mathbb{R}^{n_g \times n_g}$ are the covariance matrices of the model noise in the wave height, horizontal velocity and vertical velocity fields respectively.

Given the above computational setup, if we want to apply the optimal filter it remains to find efficient methods to calculate the weight update, $p(y_t|x_{t-1})$, and sample from the optimal proposal, $p(dx_t|x_{t-1}, y_t)$.

5.2.4 Weight calculation

From (5.9), we immediately have that

$$\log p(y_t|x_{t-1}) \propto \left(y_t - A_1 F_1(x_{t-1})\right)^T \left(A_1 \Sigma_1 A_1^T + c^2 I_{d_y}\right)^{-1} \left(y_t - A_1 F_1(x_{t-1})\right). \tag{5.14}$$

$A_1 F_1(x_{t-1})$ is the expected observation at each station given the state of the system at the previous time step and $A_1 \Sigma_1 A_1^T$ is the covariance matrix of the model noise at the observation locations.

We need to calculate (5.14) for each particle in our filter, with x_{t-1} varying over the particles. Given that we need only perform a single, offline calculation of $\left(A_1 \Sigma_1 A_1^T + c^2 I_{d_y}\right)^{-1}$, the cost of computing (5.14) for each particle and time step is $\mathcal{O}(d_y^3)$. As we expect $d_y \ll n_g$, these costs should not become prohibitive.

5.2.5 Sampling from the optimal proposal

We aim to sample from the optimal proposal (5.10). This is a Gaussian distribution with variance that can be written as

$$
\begin{aligned}
V &= (\Sigma^{-1} + A^T S^{-1} A)^{-1} \\
&= \begin{bmatrix} (\Sigma_1^{-1} + A_1^T S^{-1} A_1)^{-1} & 0_{n_g \times n_g} & 0_{n_g \times n_g} \\ 0_{n_g \times n_g} & \Sigma_2 & 0_{n_g \times n_g} \\ 0_{n_g \times n_g} & 0_{n_g \times n_g} & \Sigma_3 \end{bmatrix}.
\end{aligned} \tag{5.15}
$$

Using (5.15) and recalling the structure of F (5.11), A (5.12) and the fact that we have $S = c^2 I_{d_y}$, we can rewrite the mean of the optimal proposal as

$$
\begin{aligned}
\mu &= V(\Sigma^{-1} F(x_{t-1}) + A^T S^{-1} y_t) \\
&= \begin{bmatrix} (\Sigma_1^{-1} + A_1^T S^{-1} A_1)^{-1}(\Sigma^{-1} F(x_{t-1}) + A_1^T S^{-1} y_t) \\ (\Sigma_2^{-1} + 0_{n_g \times d_y} S^{-1} 0_{d_y \times n_g})^{-1}(\Sigma_2^{-1} F_2(x_{t-1}) + 0_{n_g \times d_y} S^{-1} y_t) \\ (\Sigma_3^{-1} + 0_{n_g \times d_y} S^{-1} 0_{d_y \times n_g})^{-1}(\Sigma_3^{-1} F_3(x_{t-1}) + 0_{n_g \times d_y} S^{-1} y_t) \end{bmatrix} \\
&= \begin{bmatrix} (\Sigma_1^{-1} + c^{-2} A_1^T A_1)^{-1}(\Sigma^{-1} F(x_{t-1}) + c^{-2} A_1^T y_t) \\ F_2(x_{t-1}) \\ F_3(x_{t-1}). \end{bmatrix}
\end{aligned} \tag{5.16}
$$

We can see from (5.15) and (5.16) that for the horizontal and vertical velocity fields we simply sample from the prior dynamics. We can therefore purely concern ourselves with sampling from the optimal distribution for the height field, $N(\mu_1, V_1)$, where

$$\begin{aligned} \mu_1 &:= (\Sigma_1^{-1} + c^{-2} A_1^T A_1)^{-1} (\Sigma^{-1} F(x_{t-1}) + c^{-2} A_1^T y_t) \\ V_1 &:= (\Sigma_1^{-1} + c^{-2} A_1^T A_1)^{-1}. \end{aligned} \tag{5.17}$$

We split the sampling from this distribution into two parts, first generating unconditional samples from the Gaussian $N(0, V_1)$; following the approach of Dietrich and Newsam [1993] and Dietrich and Newsam [1996] we can generate these samples with $\mathcal{O}(n_g \log n_g)$ cost.

Secondly, we calculate the conditional mean, μ_1, and condition our unconditional sample from the above on our observation, y_t, via the method outlined in Dietrich and Newsam [1996].

See section 5.3.2 for an outline of the approach taken in Dietrich and Newsam [1993] and Dietrich and Newsam [1996]. The focus of the papers was the simulation of conditional Gaussian random fields, we apply the methods in a particle filtering setting.

We still need to ensure we have an efficient method of calculating μ_1. We can split μ_1 into a sum using the Woodbury Matrix Identity [Higham, 2002], which states that, for appropriately sized matrices B, U, C, W, the following identity holds:

$$(B + UCW)^{-1} = B^{-1} - B^{-1}U(C^{-1} + WB^{-1}U)^{-1}WB^{-1} \tag{5.18}$$

Subbing $B = \Sigma_1^{-1}, U = A_1^T, W = c^{-2}A_1, c = I_{d_y}$ into (5.18) gives

$$\begin{aligned}(\Sigma_1^{-1} + c^{-2}A_1^T A_1)^{-1} &= \Sigma_1 - \Sigma_1 A_1^T (I_{d_y} + c^{-2}A_1\Sigma_1 A_1^T)^{-1} c^{-2} A_1 \Sigma_1 \\ &= \Sigma_1 - \Sigma_1 A_1^T (A_1 \Sigma_1 A_1^T + c^2 I_{d_y})^{-1} A_1 \Sigma_1.\end{aligned} \tag{5.19}$$

Therefore, we can write

$$\begin{aligned}\mu_1 &= \mu_{1,1} + \mu_{1,2}, \\ \mu_{1,1} &:= F_1(x_{t-1}) - \Sigma_1 A_1^T (A_1\Sigma_1 A_1^T + c^2 I_{d_y})^{-1} A_1 F_1(x_{t-1}), \\ \mu_{1,2} &:= \Big(\Sigma_1 - \Sigma_1 A_1^T (A_1 \Sigma_1 A_1^T + c^2 I_{d_y})^{-1} A_1 \Sigma_1\Big)(c^{-2} A_1^T y_t).\end{aligned} \tag{5.20}$$

As mentioned in section 5.2.4, the matrix $(A_1\Sigma_1 A_1^T + c^2 I_{d_y})^{-1}$ need only be calculated a single time, offline. Note also that we only need to compute $\mu_{1,2}$ once per time step, as this will not vary across particles.

This leaves only the computation of $\mu_{1,1}$ for each particle at each time step. Specifically we require an efficient way to calculate the matrix multiplication $\Sigma_1 v$ for the $n_g \times n_g$ covariance matrix Σ_1 and arbitrary $v \in \mathbb{R}^{n_g}$. Thankfully, we can exploit the Block Toeplitz with Toeplitz Blocks nature of Σ_1 to perform the calculation with $\mathcal{O}(n_g \log n_g)$ cost, via the Fast Fourier Transform. For more details see section 5.3.1.

5.3 Efficient computations for the optimal filter

5.3.1 Efficient matrix multiplication via Fast Fourier Transform

The one-dimensional discrete Fourier transform (DFT) of a complex-valued vector, $(v_n)_{n=0}^{N-1}$, is defined as

$$\hat{v}_k = \sum_{n=0}^{N-1} v_n e^{-2\pi ink/N}, \quad k = 0, 1, \dots, N-1. \tag{5.21}$$

Similarly, the inverse discrete Fourier transform (IDFT) is defined as

$$v_k = \frac{1}{N} \sum_{n=0}^{N-1} \hat{v}_n e^{2\pi ink/N}, \quad k = 0, 1, \dots, N-1. \tag{5.22}$$

Let $\omega = e^{-2\pi i/N}$ be the primitive N-th root of unity, then define the DFT matrix as

$$F_N := \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^1 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)(N-1)} \end{bmatrix}, \tag{5.23}$$

the one-dimensional DFT can then be expressed as $\hat{v} = F_N v$. The IDFT corresponds to the inverse of the DFT matrix and is given by

$$F_N^{-1} = \frac{1}{N} F_N^*, \tag{5.24}$$

where F_N^* is the conjugate transpose of F_N.

Computing the DFT (or IDFT) of a vector via standard matrix multiplication would come at a cost of $\mathcal{O}(N^2)$. The fast Fourier transform (FFT) [Cooley and Tukey, 1965] is an efficient algorithm for the computation of the DFT (or IDFT) that reduces this cost to $\mathcal{O}(N \log N)$ [Van Loan, 1992]. Using the FFT we can carry out matrix-vector multiplication at this reduced cost, for some specific classes of matrices.

An $N \times N$ matrix is known as circulant if all its columns contain the same elements and each column is a downwards cyclic shift of the column to its left, i.e. a matrix of the form

$$C = \begin{bmatrix} c_0 & c_{N-1} & c_{N-2} & \cdots & c_1 \\ c_1 & c_0 & c_{N-1} & \cdots & c_2 \\ c_2 & c_1 & c_0 & \cdots & c_3 \\ \vdots & \vdots & \vdots & & \vdots \\ c_{N-1} & c_{N-2} & c_{N-3} & \cdots & c_0 \end{bmatrix}. \tag{5.25}$$

So a circulant matrix, C, can be completely specified by its first column $c := (c_0, c_1, \ldots, c_{N-1})^T$.

An important property of circulant matrices is that they can be diagonalised by the DFT matrix (see e.g. Gray [2006]):

$$C = F_N^{-1} D F_N, \tag{5.26}$$

where D is a diagonal matrix containing the eigenvalues of C. Moreover, these eigenvalues can be calculated by applying the DFT matrix to the first column of C, i.e.

$$D = \text{diag}(F_N c). \tag{5.27}$$

Combining (5.26) and (5.27) we can use to DFT to carry out matrix-vector multiplication for a circulant matrix. Let v be an arbitrary N-dimensional vector, then we have

$$\begin{aligned} Cv &= F_N^{-1} D F_N v \\ &= F_N^{-1}(F_N c \circ F_N v), \end{aligned} \tag{5.28}$$

where $\circ$ denotes element-wise multiplication. So the matrix-vector multiplication, Cv, can be calculated via the DFT and IDFT - each of which can be computed efficiently using the FFT.

Circulant matrices are a special case of a broader class of matrices known as Toeplitz matrices. An $N \times N$ matrix, T, is called Toeplitz if it has the form

$$T = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-(N-1)} \\ t_1 & t_0 & t_{-1} & \cdots & t_{-(N-2)} \\ t_2 & t_1 & t_0 & \cdots & t_{-(N-3)} \\ \vdots & \vdots & \vdots & & \vdots \\ t_{N-1} & t_{N-2} & t_{N-3} & \cdots & t_0 \end{bmatrix} \tag{5.29}$$

that is, the values along each diagonal are constant. A Toeplitz matrix is completely specified by its first row and column.

An $N \times N$ Toeplitz matrix can be embedded into a $2N \times 2N$ circulant matrix with first column $(t_0, \dots, t_{N-1}, 0, t_{-(N-1)}, \dots, t_{-1})$, giving the matrix

$$T_C = \left[\begin{array}{ccccc|ccccc}
t_0 & t_{-1} & t_{-2} & \cdots & t_{-(N-1)} & 0 & t_{N-1} & t_{N-2} & \cdots & t_1 \\
t_1 & t_0 & t_{-1} & \cdots & t_{-(N-2)} & t_{-(N-1)} & 0 & t_{N-1} & \cdots & t_2 \\
t_2 & t_1 & t_0 & \cdots & t_{-(N-3)} & t_{-(N-2)} & t_{-(N-1)} & 0 & \cdots & t_3 \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
t_{N-1} & t_{N-2} & t_{N-3} & \cdots & t_0 & t_{-1} & t_{-2} & t_{-3} & \cdots & 0 \\
\hline
0 & t_{N-1} & t_{N-2} & \cdots & t_1 & t_0 & t_{-1} & t_{-2} & \cdots & t_{-(N-1)} \\
t_{-(N-1)} & 0 & t_{N-1} & \cdots & t_2 & t_1 & t_0 & t_{-1} & \cdots & t_{-(N-2)} \\
t_{-(N-2)} & t_{-(N-1)} & 0 & \cdots & t_3 & t_2 & t_1 & t_0 & \cdots & t_{-(N-3)} \\
\vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots \\
t_{-1} & t_{-2} & t_{-3} & \cdots & 0 & t_{N-1} & t_{N-2} & t_{N-3} & \cdots & t_0
\end{array}\right]. \tag{5.30}$$

If we now let v' be a $2N$-dimensional vector made by padding N zeros to an arbitrary N-dimensional vector, v, we can compute the matrix-vector multiplication $T_C v'$ efficiently via the FFT, as described above. The first N elements of $T_C v'$ will then be equal to Tv, so we can carry out matrix-vector multiplication with a Toeplitz matrix at a cost of $\mathcal{O}(N \log N)$.

Above we have shown how the 1-d DFT (and hence the FFT) can be used to efficiently carry out matrix-vector multiplication for some special classes of matrices. We go on to show how the FFT can be used to carry out the sme operation for two more classes of matrices but we must first introduce to two-dimensional DFT.

The two-dimensional DFT of a complex-valued $N \times M$ matrix, A, is defined as

$$\hat{A}_{jk} = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} e^{-2\pi inj/N} e^{-2\pi imk/M} A_{nm}, \qquad 0 \le j \le N-1, \quad 0 \le k \le M-1. \tag{5.31}$$

With the corresponding IDFT defined as

$$A_{jk} = \frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1} e^{2\pi inj/N} e^{2\pi imk/M} \hat{A}_{nm}, \qquad 0 \le j \le N-1, \quad 0 \le k \le M-1. \tag{5.32}$$

In matrix form, the 2-d DFT can be written as $\hat{A} = F_N A F_M$, where F_N and F_M are defined as in 5.23. Similarly, the matrix form of the 2-d IDFT is given by $A = F_N^{-1}\hat{A}F_M^{-1}$. These can both be computed at a cost of $\mathcal{O}(MN\log MN)$ via the FFT, applying the FFT first to the columns of A and then applying the FFT to each row of the resulting matrix.

For any $N \times M$ matrix $A = (a_{ij})$ and an arbitrary matrix B, the Kronecker product is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1M}B \\ a_{21}B & a_{22}B & \cdots & a_{2M}B \\ \vdots & \vdots & & \vdots \\ a_{N1}B & a_{N2}B & \cdots & a_{NM}B \end{bmatrix}. \tag{5.33}$$

We also define an operator that stacks the columns of a $N \times M$ matrix vertically and returns a single column of length MN.

$$\text{vec}(A) = (a_{11}, \ldots, a_{N1}, a_{12}, \ldots, a_{N2}, \ldots, a_{1M}, \ldots, a_{NM})^T. \tag{5.34}$$

The 2d-DFT can then be expressed as

$$\mathrm{vec}(\hat{A}) = (F_M \otimes F_N)\mathrm{vec}(A), \tag{5.35}$$

and similarly the 2d-IDFT can be written

$$\mathrm{vec}(A) = (F_M \otimes F_N)^{-1}\mathrm{vec}(\hat{A}). \tag{5.36}$$

An $MN \times MN$ matrix is known as block circulant with circulant blocks (BCCB) if it has the following form:

$$C = \begin{bmatrix} C_0 & C_{N-1} & C_{N-2} & \cdots & C_1 \\ C_1 & C_0 & C_{N-1} & \cdots & C_2 \\ C_2 & C_1 & C_0 & \cdots & C_3 \\ \vdots & \vdots & \vdots & & \vdots \\ C_{N-1} & C_{N-2} & C_{N-3} & \cdots & C_0 \end{bmatrix}, \tag{5.37}$$

where each C_j is an $M \times M$ circulant matrix. A BCCB matrix is fully specified by its first column $c = (c_0, \ldots, c_{N-1})^T$, where c_j is the first column of matrix C_j. All columns contain the same elements and each column is a downward cyclic shift of M places of the column to it's left.

Analogous to a circulant matrix and 1d-DFT, a BCCB matrix can be diagonalised by the 2d-DFT (see e.g. Davis [2013]). Specifically, we have

$$C = (F_N \otimes F_M)^{-1}D(F_N \otimes F_M), \tag{5.38}$$

where $D = \text{diag}((F_N \otimes F_M)c)$. Similarly to (5.28) the multiplication of an $MN \times MN$ BCCB matrix with an arbitrary MN-dimensional vector v can be computed via the 2d-DFT

$$\begin{aligned} Cv &= (F_N \otimes F_M)^{-1} D (F_N \otimes F_M) v \\ &= (F_N \otimes F_M)^{-1} ((F_N \otimes F_M) c \circ (F_N \otimes F_M) v). \end{aligned} \tag{5.39}$$

The above involves two computations of the 2d-DFT and a single computation of the 2d-IDFT, allowing the matrix-vector multiplication to be calculated at a cost of $\mathcal{O}(MN \log MN)$ via the FFT.

An $MN \times MN$ matrix is known as block Toeplitz with Toeplitz blocks (BTTB) if it has the form

$$T = \begin{bmatrix} T_0 & T_{-1} & T_{-2} & \cdots & T_{-(N-1)} \\ T_1 & T_0 & T_{-1} & \cdots & T_{-(N-2)} \\ T_2 & T_1 & T_0 & \cdots & T_{-(N-3)} \\ \vdots & \vdots & \vdots & & \vdots \\ T_{N-1} & T_{N-2} & T_{N-3} & \cdots & T_0 \end{bmatrix}, \tag{5.40}$$

the values along each block diagonal are constant and each T_j is an $M \times M$ Toeplitz matrix.

We wish to efficiently compute the matrix-vector multiplication Tv, for an arbitrary vector v of length MN. We know that we can do this for a BCCB matrix, so similar to the case of a Toeplitz matrix, we want to embed our BTTB matrix inside a BCCB matrix.

Each Toeplitz block can be embedded inside a $2M \times 2M$ circulant block as in (5.30), which gives a block Toeplitz matrix with circulant blocks. The resulting matrix can then be embedded inside a $4MN \times 4MN$ BCCB matrix in an almost identical way to (5.30), with the circulant blocks taking the place of single elements in the matrix. Denote this BCCB matrix by T_{BC}.

Now consider the vector v to be made up of N smaller vectors of length M. Extend v by first appending M zeros to each of the smaller vectors and then appending an additional N zero vectors of length $2M$ to the larger vector. So for a vector $v = (v_0 \quad v_1 \quad \cdots \quad v_N)^T$ where each v_j is a vector of length M, the corresponding extended vector is

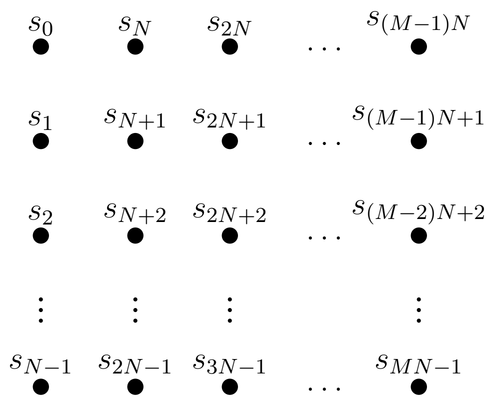
$$v' = (v_0 \quad 0_M \quad v_1 \quad 0_M \quad \cdots \quad v_N \quad 0_M \quad 0_{2NM})^T, \tag{5.41}$$

where 0_M denotes a vector of zeros of length M. The BCCB matrix vector multiplication $T_{BC}v'$ can now be computed efficiently via FFT as described in (5.39). The result Tv can now be obtained from the resulting vector by taking the first M elements of the first N sub-vectors of length M of the resulting vector [Vogel, 2002], i.e. denote $T_{BC}v' = (\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_{4NM-1})^T$, where each $\hat{v}$ is a single element. Then we have

$$Tv = (\hat{v}_0, \ldots, \hat{v}_{M-1}, \hat{v}_{2M}, \ldots, \hat{v}_{3M-1}, \cdots, \hat{v}_{M(2N-2)}, \ldots, \hat{v}_{M(2N-1)-1}). \tag{5.42}$$

Recall that we want to be able to compute $\Sigma_1 v$ efficiently in order to perform the calculations in (5.20), where Σ_1 is the $n_g \times n_g$ covariance matrix of a stationary Gaussian random field and $v \in \mathbb{R}^{n_g}$. Above we have shown that if Σ_1 is a BTTB matrix then we can exploit its structure to perform the computation at a cost of $\mathcal{O}(n_g \log n_g)$.

It is actually the case that any stationary Gaussian field on a grid of points that are regularly spaced in the x and y directions has a covariance matrix with a BTTB structure. Suppose we have a stationary Gaussian field, Z, on a set of points $\{s_0, s_1, \ldots, s_{MN-1}\}$ arranged in a regular grid as shown below:



As Z is stationary, $\mathbb{E}(Z(s_j))$ is a constant, without loss of generality we may assume that $\mathbb{E}(Z(s_j)) = 0$. Also by stationarity we have that the covariance between points s_i and s_j depends only on $s_i - s_j$. So we have

$$\begin{aligned}\text{Cov}(Z(s_i), Z(s_j)) :&= \mathbb{E}\left[(Z(s_i) - \mathbb{E}(Z(s_j))\,(Z(s_i) - \mathbb{E}\left(Z(s_j)\right)\right] \\ &= \mathbb{E}\left[Z(s_i)(Z(s_j)\right] \\ &= r(x_i - x_j, y_i - y_j),\end{aligned} \tag{5.43}$$

where (x_i, y_i), (x_j, y_j) are the x and y coordinates of s_i and s_j respectively and $r : \mathbb{R}^2 \to \mathbb{R}$ is a covariance function. Let R be the $MN \times MN$ covariance matrix for $[Z(s_1), Z(s_2) \ldots, Z_(s_{MN-1})]$ such that

$$R(i,j) = \mathsf{Cov}(Z(s_i), Z(s_j)) = r(x_i - x_j, y_i - y_j). \tag{5.44}$$

This covariance matrix has the form

$$R = \begin{bmatrix} R_{00} & R_{01} & R_{02} & \cdots & R_{0,M-1} \\ R_{10} & R_{11} & R_{12} & \cdots & R_{1,M-1} \\ R_{20} & R_{21} & R_{22} & \cdots & R_{2,M-1} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{M-1,0} & R_{M-1,2} & R_{M-1,3} & \cdots & R_{M-1,M-1} \end{bmatrix}, \tag{5.45}$$

where each R_{mn} is an $N \times N$ submatrix that corresponds to the covariance between points in the mth and nth column of points when the points are arranged in a regular grid. Let $(R_{mn})_{kl}$ denote the (k, l)th element of the (m, n)th submatrix.

Without loss of generality we can also assume that adjacent points in the grid are separated by unit distance in both the x and y directions. Consider an element $(R_{mn})_{kl}$ and an integer a such that $(R_{mn})_{k+a,l+a}$ is an element of the submatrix R_{mn}, then we have

$$\begin{aligned}
(R_{mn})_{k+a,l+a} &= \text{Cov}(s_{Mm+k+a}, s_{Mn+l+a}) \\
&= r(x_{Mm+k+a} - x_{Mn+l+a}, y_{Mm+k+a} - y_{Mn+l+a}) \\
&= r(x_{Mm+k} - x_{Mn+l}, y_{Mm+k+a} - y_{Mn+l+a}) \\
&= r(x_{Mm+k} - x_{Mn+l}, y_{Mm+k} + a - (y_{Mn+l} + a)) \\
&= r(x_{Mm+k} - x_{Mn+l}, y_{Mm+k} - y_{Mn+l}) \\
&= (R_{mn})_{kl},
\end{aligned} \tag{5.46}$$

so each $N \times N$ submatrix R_{mn} is Toeplitz. Similarly consider a submatrix R_{mn} and an integer b such that $R_{m+b,n+b}$ is also a submatrix of R, then we have

$$\begin{aligned}
(R_{m+b,n+b})_{kl} &= \text{Cov}(s_{M(m+b)+k}, s_{M(n+b)+l}) \\
&= r(x_{Mm+Mb+k} - x_{Mn+Mb+l}, y_{Mm+Mb+k} - y_{Mn+Mb+l}) \\
&= r(x_{Mm+Mb+k} - x_{Mn+Mb+l}, y_{Mm+k} - y_{Mn+l}) \\
&= r(x_{Mm+k} + Mb - (x_{Mn+l} + Mb), y_{Mm+k} - y_{Mn+l}) \\
&= r(x_{Mm+k} - x_{Mn+l}, y_{Mm+k} - y_{Mn+l}) \\
&= (R_{mn})_{kl},
\end{aligned} \tag{5.47}$$

implying that R is block Toeplitz and therefore R is also BTTB.

5.3.2 Efficient sampling of Gaussian Random Fields

We consider the problem of generating realisations of a stationary Gaussian random field on a rectangular grid, conditioned on direct or indirect observations. We follow the approach outlined in Dietrich and Newsam [1996], which extends the methods of Dietrich and Newsam [1993] (for generating unconditional realisations) to the conditional case.

Specifically, we seek to generate realisations of a stationary Gaussian random field, $Z(x, y)$, on a 2D-domain, $\Omega = [0, P] \times [0, Q]$, discretised to form a grid, $\Omega_1 = \{(x_i, y_j) = (i\delta_x, j\delta_y) : 0 \leqslant i \leqslant p, 0 \leqslant j \leqslant q\}$, of size $n_1 = (p + 1) \times (q + 1)$, where δ_x and δ_y are the distance between nodes of Ω_1, in the x and y directions respectively. These are to be conditioned on n_2 measurements of Z on a grid $\Omega_2 = \{(x_k, y_k) : 1 \leqslant k \leqslant n_2\}$.

As Z is stationary, its mean is constant, without loss of generality we assume the mean of Z equals zero. Then if z_1 and z_2 are vectors of samples of Z on Ω_1 and Ω_2 respectively, then

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}, \tag{5.48}$$

has zero mean and covariance matrix

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \tag{5.49}$$

where R_{ij} is the covariance matrix of Z between points in Ω_i and Ω_j.

It can be shown (see e.g. Eaton [1983]) that the distribution of z_1 conditioned on a direct observation $z_2^\star$ of z_2 is Gaussian with mean

$$\mu_{z_1|z_2^\star} = R_{12}R_{22}^{-1}z_2^\star, \tag{5.50}$$

and covariance matrix

$$R_{z_1|z_2^\star} = R_{11} - R_{12}R_{22}^{-1}R_{21}. \tag{5.51}$$

If the random variable, z, is distributed $N(0, R)$ (as in (5.48)), on $\Omega_1 \cup \Omega_2$, then

$$z_1|z_2^* = \mu_{z_1|z_2^\star} + z_1 - R_{12}R_{22}^{-1}z_2 \tag{5.52}$$

is distributed according to $N(\mu_{z_1|z_2^\star}, R_{z_1|z_2^\star})$, the desired distribution for a simulation of the Gaussian field Z on Ω_1, conditioned on direct measurements of Z on Ω_2.

Often in practice the observations, $z_2^\star$, are observed with some observation error. Assuming this observations error is Gaussian with correlation matrix Σ, as in (5.6), this can be accounted for by replacing R_{22} in the above with the matrix $R_{22} + \Sigma$.

Many methods that compute conditional simulations via the above require the observation grid, Ω_2, to be a subset of Ω_1 and so the points in Ω_2 are perturbed to coincide with points in Ω_1. Though this should only result in a small change in R_{12} and R_{22}, we can see from equation (5.52) that the conditional realisations depend on R_{22}^{-1}, which may be sensitive to small perturbations of Ω_2. This limitation was the motivation behind Dietrich and

Newsam [1996], which extends the results of Dietrich and Newsam [1993] to allow efficient simulation of Gaussian fields conditioned on some direct or indirect observations. These methods are outlined below.

We have a covariance function, r, defined on $\Omega = [0, P] \times [0, Q]$, extend this function to a new function, $\bar{r}$, defined on $[0, 2P] \times [0, 2Q]$, in the following way:

$$\begin{aligned} \bar{r}(x, y) &= r(x, y) && 0 \leqslant x \leqslant P,\ 0 \leqslant y \leqslant Q \\ \bar{r}(x, y) &= r(2P - x, y) && P \leqslant x \leqslant 2P,\ 0 \leqslant y \leqslant Q \\ \bar{r}(x, y) &= r(x, 2Q - y) && 0 \leqslant x \leqslant P,\ Q \leqslant y \leqslant 2Q \\ \bar{r}(x, y) &= r(2P - x, 2Q - y) && P \leqslant x \leqslant 2P,\ Q \leqslant y \leqslant 2Q \end{aligned} \tag{5.53}$$

We then extend $\bar{r}$ to the entire plane and assume that it is a covariance function for a zero-mean, stationary, periodic random field, which contains a single period in $[0, 2P] \times [0, 2Q]$, denoted by $\bar{Z}(x, y)$. Also extend Ω_1, the discretisation of Ω, to cover $[0, 2P] \times [0, 2Q]$ in the natural way, denote this extension by $\bar{\Omega}_1$. Also denote

$$\bar{z} = \begin{bmatrix} \bar{z}_1 \\ z_2 \end{bmatrix} \tag{5.54}$$

to be a vector of samples of $\bar{Z}(x, y)$ on $\bar{\Omega}_1 \cup \Omega_2$. Let $\bar{R}$ be the covariance matrix of $\bar{Z}$, it can be partitioned as

$$\bar{R} = \begin{bmatrix} \bar{R}_{11} & \bar{R}_{12} \\ \bar{R}_{21} & R_{22} \end{bmatrix}, \tag{5.55}$$

where $\bar{R}_{11}$ is the covariance matrix between pairs of points in $\bar{\Omega}_1$, $\bar{R}_{12}$ is the covariance matrix between points in $\bar{\Omega}_1$ and Ω_2 and $\bar{R}_{21} = \bar{R}_{12}^T$. Restricting $\bar{r}$ to $[0, P] \times [0, Q]$ is exactly R, so the restriction of a sample $\bar{z}$ to $\Omega_1 \cup \Omega_2$ has R as it's covariance matrix. Therefore, once a sample, $\bar{z}$, is computed over the extended grid $\bar{\Omega}_1 \cup \Omega_2$ we can restrict this to $\Omega_1 \cup \Omega_2$ to yield a realisation of a sample z, as desired.

To compute a realisation of $\bar{z}$ we construct a square root,$\bar{R}^{1/2}$ of $\bar{R}$ with the property $\bar{R}^{1/2}(\bar{R}^{1/2})^* = \bar{R}$ such that a matrix-product $\bar{R}^{1/2}\epsilon$ can be computed efficiently for any complex-valued vector ϵ. If $\epsilon \sim N(0, I)$, then a realisation $\bar{z} = \bar{R}^{1/2}\epsilon$ will have R as it's covariance matrix and be drawn from the desired distribution.

Thanks to the stationarity of $\bar{Z}(x, y)$, the periodicity of $\bar{r}$ and the fact that the points in $\bar{\Omega}_1$ are equispaced in the x and y directions, the matrix $\bar{R}_{11}$ is circulant. Therefore, $\bar{R}_{11}$ is uniquely determined by its first column vector $\bar{\rho}$ and can be decomposed as

$$\bar{R}_{11} = W \Lambda W^*, \tag{5.56}$$

where W is the normalised DFT matrix and Λ is a diagonal matrix with entries $(4pq)^{1/2} W \bar{\rho}$.

It is then easy to check that

$$\bar{R}^{1/2} = \begin{bmatrix} W\Lambda^{1/2} & 0 \\ K & L \end{bmatrix}, \tag{5.57}$$

where $K = \bar{R}_{21} W \Lambda^{-1/2}$ and L is a matrix such that $LL^T = \bar{R}_{21}\bar{R}_{11}^{-1}\bar{R}_{12}$ (e.g. as a result of a Cholesky decomposition), is a square root of $\bar{R}$.

Given $\bar{R}^{1/2}$ and $\epsilon \sim N(0, I)$ a complex normal random variable, then the real and imaginary parts of $\bar{R}^{1/2}\epsilon$ are two independent realisations of $\bar{z}$ and restricting these to $\Omega_1 \cup \Omega_2$ gives two independent realisations of z.

Rewriting $\bar{z} = \bar{R}^{1/2}\epsilon$ gives

$$\bar{z} = \begin{bmatrix} \bar{z}_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} W\Lambda^{1/2} & 0 \\ K & L \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \end{bmatrix}. \tag{5.58}$$

For a grid of size $\bar{n}_1 = 4pq$ and n_2 observations the setup computational costs of the above are as follows.

- Λ can be computed at a cost of $\mathcal{O}(\bar{n}_1 \log \bar{n}_1)$ via FFT.
- The computation of $K^* = \Lambda^{-1/2} W^* \bar{R}_{12}$ is dominated by computing the inverse FFT of each column of $\bar{R}_{12}$ at a cost of $\mathcal{O}(\bar{n}_1 n_2 \log(\bar{n}_1))$.
- To compute L, KK^* is computed at a cost of $\mathcal{O}(\bar{n}_1 n_2^2)$ followed by computing $R_{22} - KK^*$ and its Cholesky decomposition at a cost of $\mathcal{O}(n_2^3)$.

We have $\bar{n}_1 \approx 4n_1$, assuming $n_2 \ll n_1$ the total setup cost is $\mathcal{O}(n_1 n_2 \max(\log n_1, n_2))$.

Once the setup is complete each realisation involves calculating $\bar{z}_1 = W\Lambda\epsilon_1$ via FFT at a cost of $\mathcal{O}(n_1 \log n_1)$ and $z_2 = K\epsilon_1 + L\epsilon_2$ at a cost of $\mathcal{O}(n_1 n_2)$. So the total computational cost is a setup cost of $\mathcal{O}(n_1 n_2 \max(\log n_1, n_2))$ and a cost of $\mathcal{O}(n_1 \max(\log n_1, n_2))$ for computing each realisation.

It is important to note that the above method is only valid if $\bar{R}$ is a positive definite matrix, without this condition it is not possible to find a square root of $\bar{R}$ such that $\bar{R}^{1/2}(\bar{R}^{1/2})^* = \bar{R}$.

To ensure $\bar{R}$ is positive definite we need to find conditions such that the periodic extension, $\bar{r}$, of a positive definite covariance function r, is also positive definite. For details on conditions required for this see Dietrich and Newsam [1996].

5.4 Discrete-time model noise experiments

We conduct numerical experiments to demonstrate the performance of the optimal filter compared to the bootstrap algorithm. The following parameters were selected to create a challenging but computationally feasible test scenario that exhibits the key features of tsunami wave propagation:

- 2D-domain $\Omega = [0, P] \times [0, Q]$ with $P = Q = 40000$, discretised as in 5.2, with $p = q = 100$. This 40km × 40km domain with 400m resolution provides sufficient space for wave propagation while maintaining computational tractability.
- Constant sea depth of $D(x, y) = 1500\text{m}$, representing a simplified deep-ocean environment.

- Boundary damping factor $\lambda = 0.015$, with a boundary size of 15 grid points, chosen to minimise boundary reflections without have a significant impact on the dynamics of the propagating wave.

- Independent observation noise across stations, distributed $N(0, c^2)$, with $c = 0.01$m. This noise level was chosen to be small enough to allow accurate tracking while large enough to present a meaningful filtering challenge.

- Model noise independent across height and horizontal and vertical velocity fields, with covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ respectively. These covariance matrices correspond to the covariance function

$$r(x, y) = \sigma^2 \exp\left(-\sqrt{\left(\frac{x^2 + y^2}{l^2}\right)}\right). \tag{5.59}$$

  where we set the correlation length $l = 3$km for all three fields, selected to represent spatial correlations on the scale of tsunami wavelengths. We use $\sigma = 0.1$m for the covariance of the height field Σ_1 and $\sigma = 1$m for the covariance of the velocity fields, Σ_2, Σ_3.

- Initialise the wave at the point $(10000, 10000)$, with an initial spread $s = 3000$m and initial max height, $h_{\text{max}} = 3$m, creating a moderately sized test wave.

- For both the bootstrap and optimal filters we use 100 particles, a number chosen to demonstrate filter behaviour while keeping computational costs reasonable for multiple experimental runs. Resampling is activated when the ESS drops below 50, following common practice in the particle filtering literature.

- Simulation run for 240 seconds, with observations every second from 70 seconds onwards, allowing the wave to develop before data assimilation begins.

We then run experiments with four different sets of observation station locations:

- A square of 36 stations, centred in the middle of our grid, with side lengths of approximately 16km. Results for this experiment can be seen in figure 5.2.
- A set of stations randomly placed with the restrictions that they may not be placed in the square of side length 16km with its lower left corner at the origin, and may not be placed with either the x or y coordinate greater than 32km. Results for this experiment can be seen in figure 5.3.
- Three sets of 10 stations equally spaced across radii 20km, 28km and 36km from the origin. Results for this experiment can be seen in figure 5.4.
- Three sets of 10 stations equally spaced across radii 20km, 22km and 24km from the origin. Results for this experiment can be seen in figure 5.5.

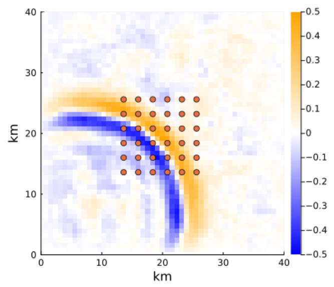
Figures 5.2, 5.3, 5.4 and 5.5 show comparisons of the actual wave height to the wave height estimated via the posterior mean of the optimal filter and bootstrap filter, for different arrangements of station locations. Also shown is the effective sample size of the optimal filter over time.

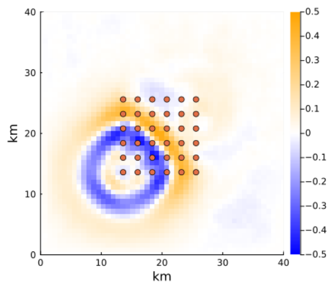
In each case the optimal filter produces a more accurate representation of the wave height than the bootstrap filter, as we would expect. Though the performance of the optimal filter seems to be highly dependent on the location of the observation stations. Compare the regular grid arrangement of figure 5.2 and the random arrangement of figure 5.3, it's clear that the more spatially coherent information provided by the regular grid allows the filter to more accurately reconstruct the wavefield than the randomly positioned stations. Comparing the concentric ring placements of figure 5.4 and figure 5.5 shows that the spacing of the stations is also an important factor in determining how well the optimal filter can reconstruct the wave. A potential use of experiments like these is in potentially aiding the future placement of sensors for detecting future tsunamis, and while a full study is beyond the scope of this thesis, these early results suggest that strategic regular placement would be a good starting point.

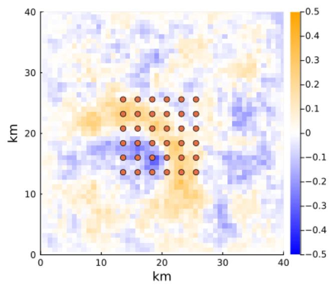
In all cases the effective sample size of the optimal filter does not collapse, indicating some stability in the particle weights and avoiding frequent resampling. This should ensure that particle diversity remains high throughout. This is in contrast to the bootstrap filter where at almost all time steps the particle weight collapses down to a single particle and resampling is triggered.

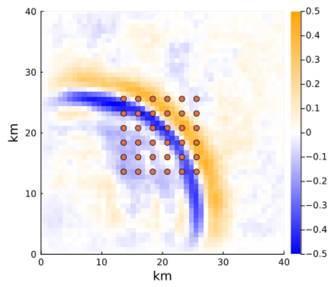
We again see that the placement of the observation stations affects the performance of the optimal filter. The random placement in figure 5.3 provides less spatially coherent information than the regular grid in figure 5.2, resulting in the propagated state of the being likely to differ greatly from the observations. This discrepancy leads to a higher variance in the importance weights, this causes the ESS to collapse more frequently, triggering the resampling step. This comparison demonstrates the link between the quality of the observation network and the computational stability of the optimal filter

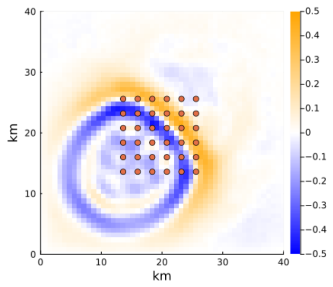
Figures 5.6 (along with plots in Appendix B) provides a more granular analysis of the optimal filter's performance, they show the estimated probability density functions of the wave height at all observation stations at different time points in the case where stations are on a square grid in the centre of the domain. The actual and observed height are also shown. A successful filter should produce a posterior density that is centred near the observation but is broad enough to assign high probability to the true state. As can be seen across the subplots, the optimal filter consistently achieves this. The true height (black line) regularly falls within the high-probability region of the estimated density, demonstrating that the filter is correctly balancing the information from the noisy data with the uncertainty from the model to produce a reliable posterior distribution.

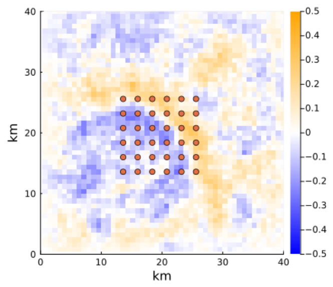

(a) $t = 140s$, true signal

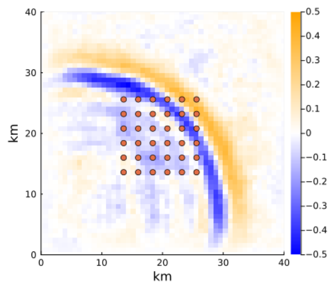

(b) $t = 140s$, optimal filter

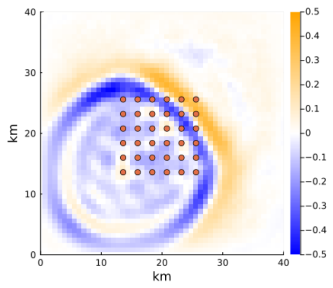

(c) $t = 140s$, bootstrap

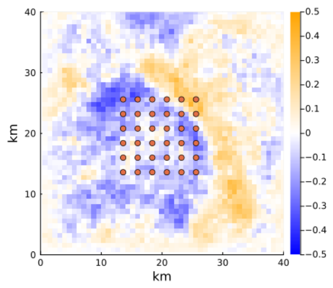

(d) $t = 170s$, true signal

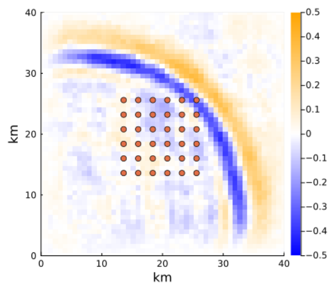

(e) $t = 170s$, optimal filter

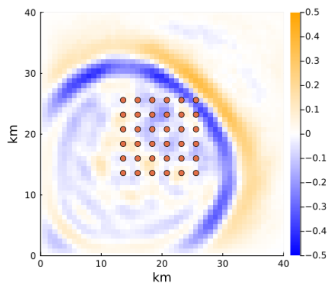

(f) $t = 170s$, bootstrap

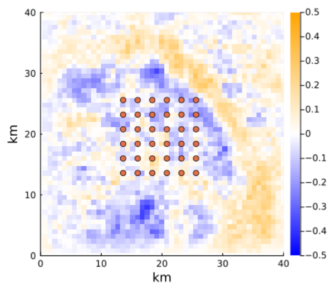

(g) $t = 200s$, true signal

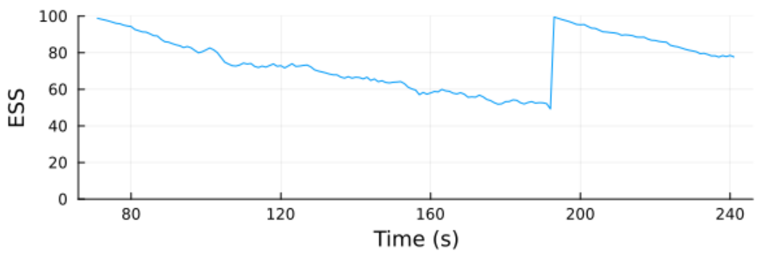

(h) $t = 200s$, optimal filter


(i) $t = 200s$, bootstrap


(j) $t = 230s$, true signal

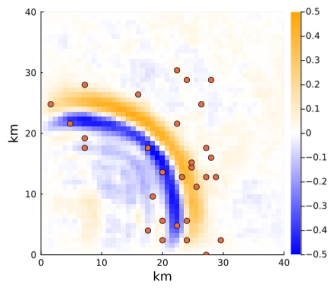

(k) $t = 230s$, optimal filter

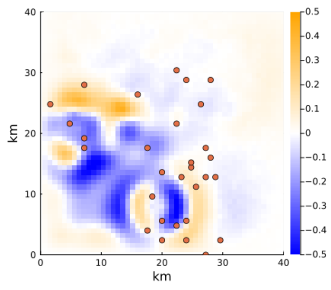

(l) $t = 230s$, bootstrap

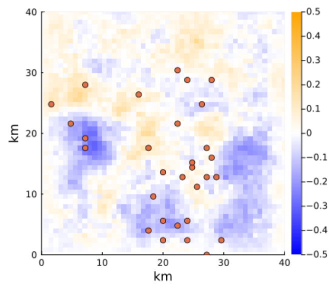


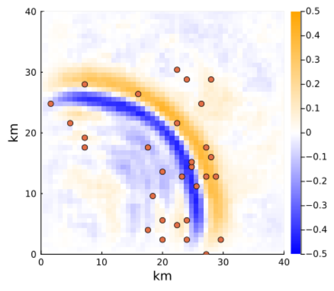
(m) Optimal filter effective sample size

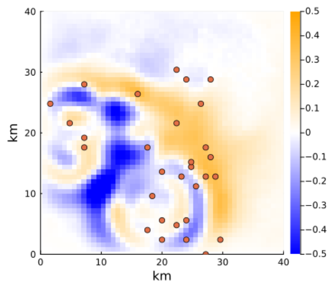
Figure 5.2: Estimates of wave height via posterior mean of bootstrap and optimal filters compared with the actual wave heights. Stations are placed in a regular grid at the centre of the domain.

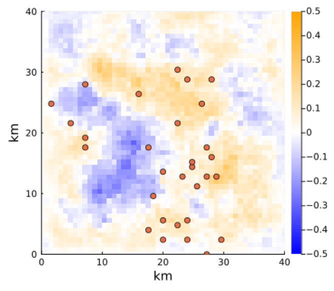

(a) $t = 140s$, true signal

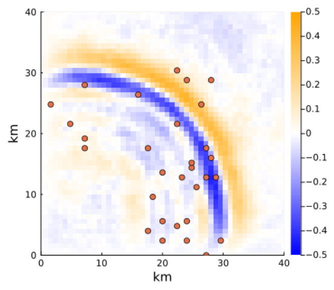

(b) $t = 140s$, optimal filter

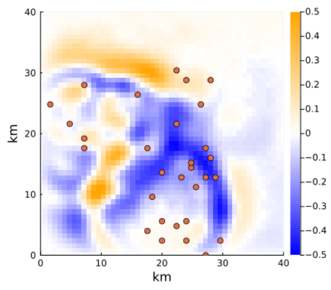

(c) $t = 140s$, bootstrap

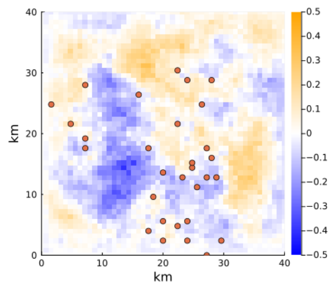

(d) $t = 170s$, true signal

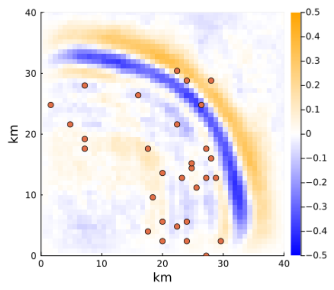

(e) $t = 170s$, optimal filter

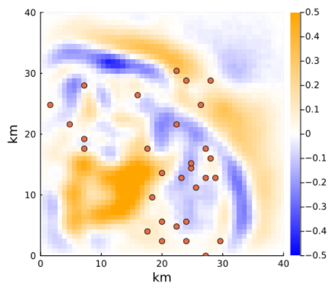

(f) $t = 170s$, bootstrap

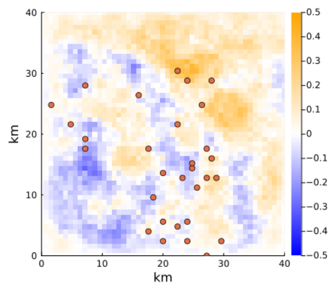

(g) $t = 200s$, true signal

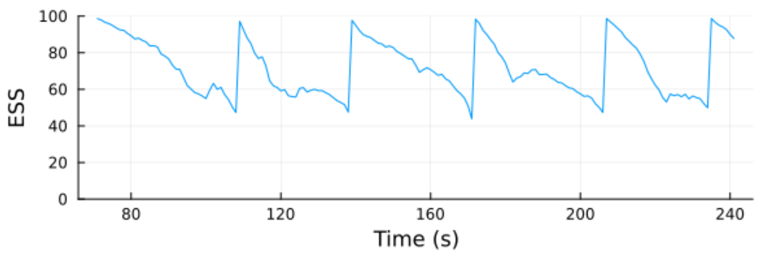

(h) $t = 200s$, optimal filter


(i) $t = 200s$, bootstrap


(j) $t = 230s$, true signal

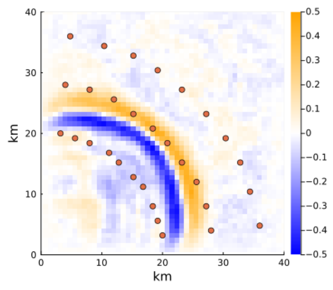

(k) $t = 230s$, optimal filter

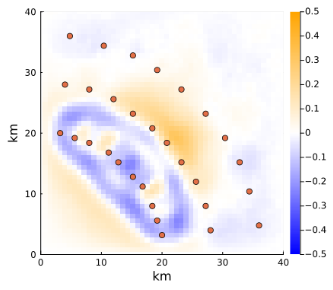

(l) $t = 230s$, bootstrap

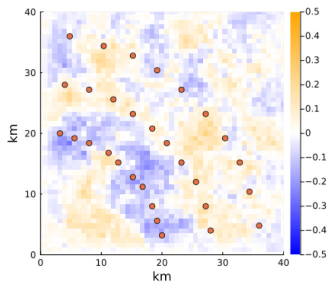


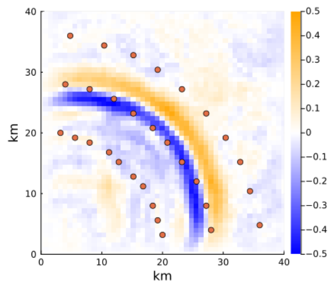
(m) Optimal filter effective sample size

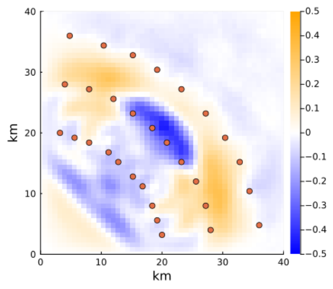
Figure 5.3: Estimates of wave height via posterior mean of bootstrap and optimal filters compared with the actual wave heights. Stations are placed randomly in a subset of the domain.

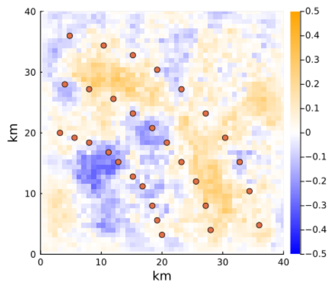

(a) $t = 140s$, true signal

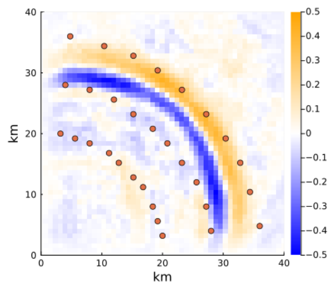

(b) $t = 140s$, optimal filter

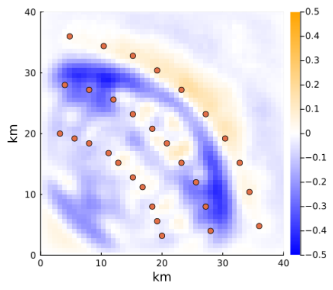

(c) $t = 140s$, bootstrap

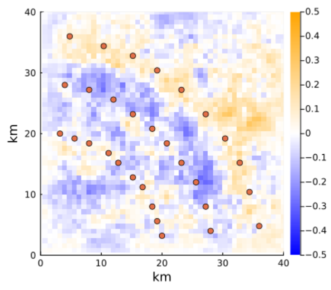

(d) $t = 170s$, true signal

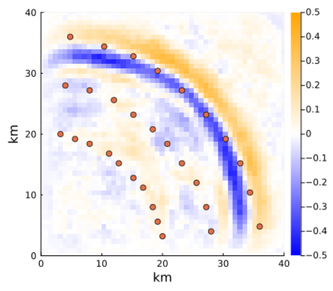

(e) $t = 170s$, optimal filter

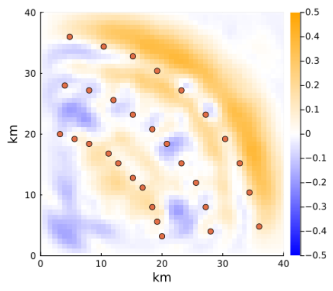

(f) $t = 170s$, bootstrap

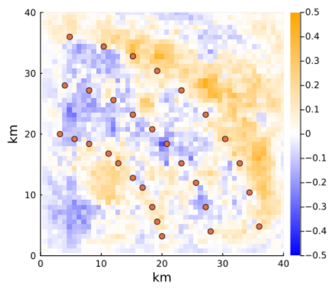

(g) $t = 200s$, true signal

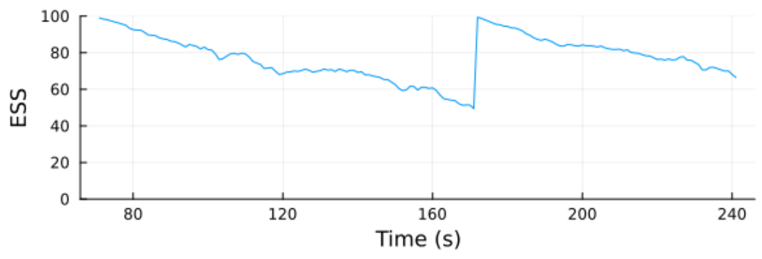

(h) $t = 200s$, optimal filter


(i) $t = 200s$, bootstrap


(j) $t = 230s$, true signal

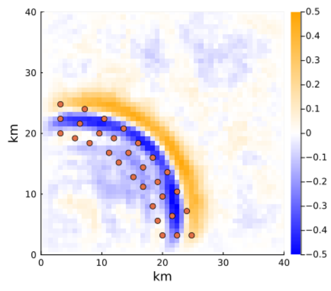

(k) $t = 230s$, optimal filter

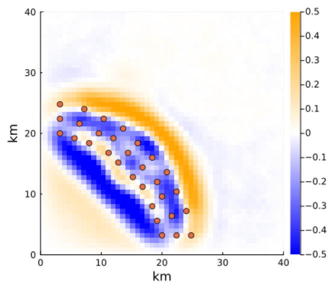

(l) $t = 230s$, bootstrap

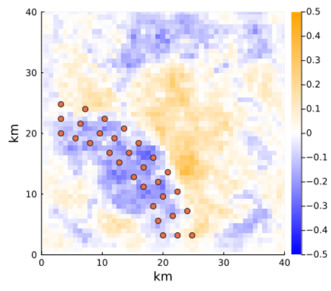


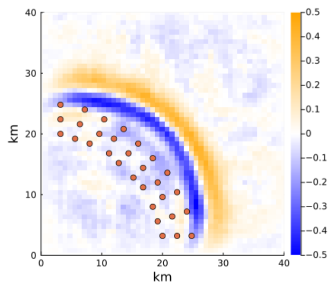
(m) Optimal filter effective sample size

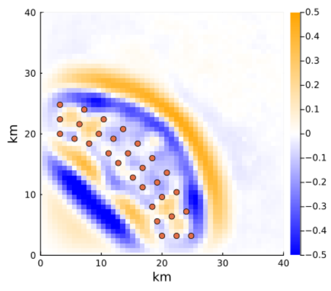
Figure 5.4: Estimates of wave height via posterior mean of bootstrap and optimal filters compared with the actual wave heights. There are three sets of 10 stations equally spaced across radii 20km, 28km and 36km from the origin.

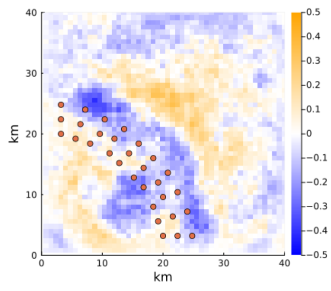

(a) $t = 140s$, true signal

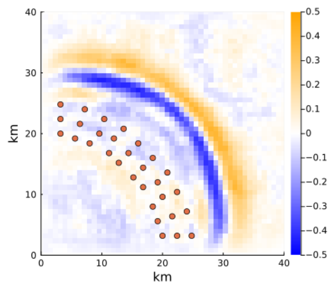

(b) $t = 140s$, optimal filter

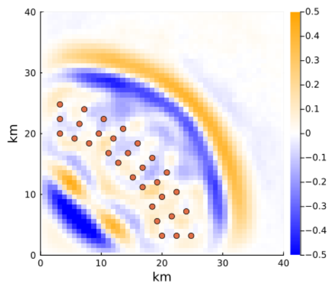

(c) $t = 140s$, bootstrap

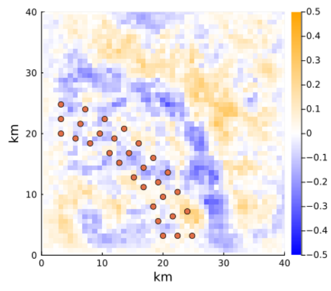

(d) $t = 170s$, true signal

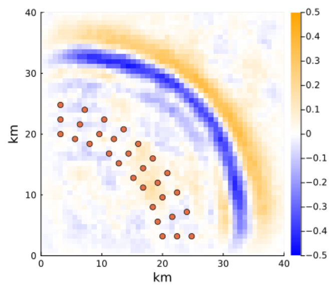

(e) $t = 170s$, optimal filter

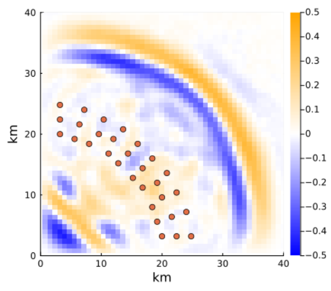

(f) $t = 170s$, bootstrap

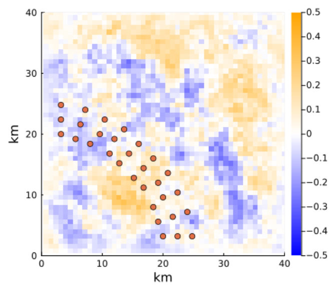

(g) $t = 200s$, true signal

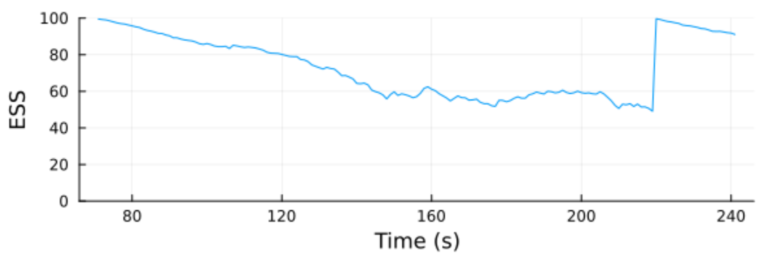

(h) $t = 200s$, optimal filter


(i) $t = 200s$, bootstrap


(j) $t = 230s$, true signal


(k) $t = 230s$, optimal filter


(l) $t = 230s$, bootstrap



(m) Optimal filter effective sample size

Figure 5.5: Estimates of wave height via posterior mean of bootstrap and optimal filters compared with the actual wave heights. There are three sets of 10 stations equally spaced across radii 20km, 22km and 24km from the origin.

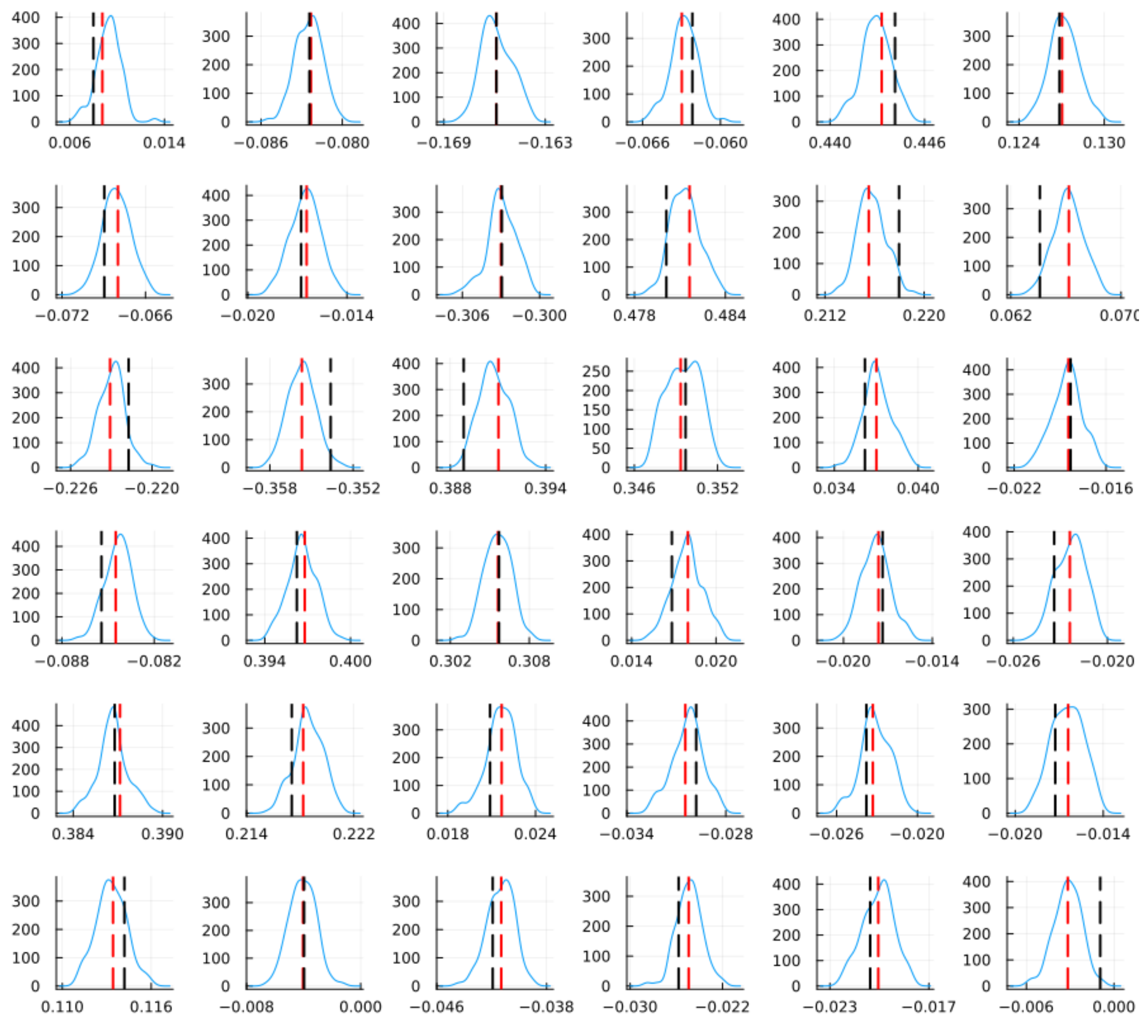


Figure 5.6: Estimated probability density functions of wave height via the optimal filter at all station locations at 140 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

Chapter 6

A particle filter for a Tsunami SPDE model

In this section we consider the filtering problem for a tsunami system, modelled as a stochastic partial differential equation (SPDE) with noisy observations of the wave height at discrete time steps. Specifically, we model the wave on a two-dimensional domain as

$$\begin{aligned}
\frac{\partial \eta(x,y,t)}{\partial t} &= -\frac{\partial M(x,y,t)}{\partial x} - \frac{\partial N(x,y,t)}{\partial y} + \Sigma_1^{\frac{1}{2}} W(x,y,t) \\
\frac{\partial M(x,y,t)}{\partial x} &= -gD(x,y)\frac{\partial \eta(x,y,t)}{\partial x} + \Sigma_2^{\frac{1}{2}} W(x,y,t) \\
\frac{\partial N(x,y,t)}{\partial y} &= -gD(x,y)\frac{\partial \eta(x,y,t)}{\partial y} + \Sigma_3^{\frac{1}{2}} W(x,y,t),
\end{aligned} \tag{6.1}$$

where $\eta(x,y,t)$ is the wave height, $M(x,y,t)$ and $N(x,y,t)$ are the horizontal and vertical velocities respectively, $D(x,y)$ is the depth of the sea at the point (x,y), g is the acceleration due to gravity, $W(x,y,t)$ is a space-time white noise process and $\Sigma_1, \Sigma_2, \Sigma_3$ are operators corresponding to some covariance functions.

The dynamics described by the SPDE (6.1) present a more complex filtering problem compared to the SDEs discussed in earlier sections. Unlike the SDE case, where the system states are represented as finite-dimensional vectors in $\mathbb{R}^d$, the states here are functions of both space and time, residing within an infinite-dimensional Hilbert space. Additionally, while the previous SDE model featured d independent one-dimensional Brownian motions as the driving noise, the SPDE model involves space-time white noise, a more complex, infinite-dimensional process with spatial correlations.

When simulating (5.1), it is necessary to discretise the system in both time and space, unlike the SDE case where only time discretisation was required. This added complexity poses a challenge for filtering algorithms because spatial discretisation can lead to high-dimensional state spaces, potentially resulting in particle weight degeneracy and poor filtering distribution estimates. Therefore, an effective particle filtering algorithm for this setting must accommodate high-dimensional state spaces, handle complex noise structures, and be robust to varying levels of time and space discretisation.

This setting also poses significant computational challenges. Solving the SPDE (6.1) involves applying numerical methods to both solve the PDE and simulate the space-time white noise process, which becomes increasingly computationally intensive as the spatial and temporal resolution of the simulation increases. For very fine resolutions, running a particle filter with a large number of particles may be infeasible due to high costs. Thus, a trade-off must be found between finer spatial and temporal resolutions, which reduce discretisation error, and the overall computational cost of the simulations and particle filtering.

Additionally, calculating the Radon-Nikodym derivative between a proposal and target distribution in this context is more computationally demanding compared to cases with independent Brownian motion noise. This is because it involves multiplying a vector by the inverse of a large covariance matrix. For independent Brownian motions, this matrix is diagonal, and the multiplication scales linearly with the number of dimensions of the state space. However, with space-time white noise, the covariance matrix may be dense, causing the multiplication to scale quadratically with the number of dimensions when performed naively.

Memory usage may also be of concern in this setting. The state space of the system is represented as a grid of wave heights, horizontal velocities, and vertical velocities. As the number of grid points increases, the memory required to store the state space and particle weights grows, potentially leading to memory limitations for very fine spatial resolutions.

These concerns further motivate the development of a particle filtering algorithm that can effectively handle high-dimensional state spaces, complex noise structures, and varying levels of spatial and temporal discretisation, without the need for a prohibitively large number of particles. In the following sections, we propose an algorithm that addresses these challenges and demonstrate its effectiveness in filtering the tsunami system described by the SPDE (6.1).

6.1 Model discretisation and setup

From this point, we work with a discretised approximation of the SPDE (6.1) on a 2D grid of size $n_g = (p+1) \times (q+1)$, where p and q are the number of grid points in the x and y directions, respectively. We represent the state of the system as a vector $X_t \in \mathbb{R}^{3n_g}$, mapping the wave heights, horizontal velocities, and vertical velocities to their corresponding entries. The system dynamics are then given by

$$dX_t = F(X_t)dt + \Sigma^{\frac{1}{2}} dW_t, \tag{6.2}$$

where F is considered as a mapping $F : \mathbb{R}^{3n_g} \to \mathbb{R}^{3n_g}$, as in (5.11) and corresponds to the deterministic dynamics of the shallow water equation as defined in (5.1), solved via the finite difference scheme described in 5.1.2. The covariance matrix Σ is block diagonal, with blocks corresponding to the covariance matrices of the height, horizontal velocity, and vertical velocity fields, respectively - as in (5.13).

We observe the system at discrete time steps as

$$Y_k = AX_{t_k} + \zeta_t, \tag{6.3}$$

where A is a linear operator and $\zeta_t \sim N(0, S)$, for some covariance matrix S.

We again assume we have d_y observation stations located at fixed points on our 2D grid, and that we have only observations of the wave height. The operator A selects these points from our grid and can be represented by a $d_y \times 3n_g$ matrix, as in (5.12).

Observational noise is again assumed to be independent across stations and each follows a Gaussian distribution with mean 0 and variance c^2.

Simulation of the SPDE is done via a discretisation in time, using the Euler-Maruyama method,

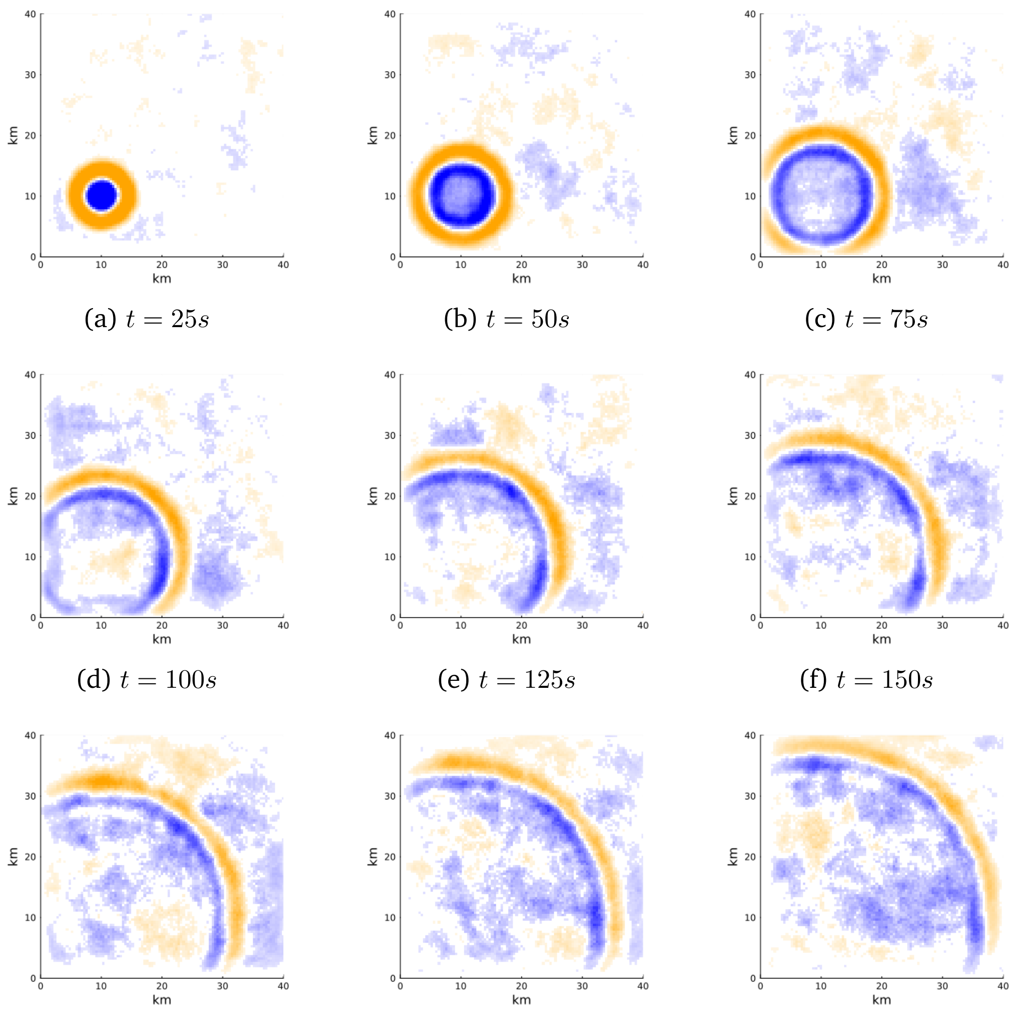
$$X_{t+\Delta t} = X_t + F(X_t)\Delta t + \sqrt{\Delta t}\eta \tag{6.4}$$

where $\eta \sim N(0, \Sigma)$. Theoretically the Brownian increment η corresponds to a realisation of a Gaussian random field - an element of an infinite-dimensional Hilbert space. To work with and sample from this distribution, we must discretise the space using a regular grid, as described in the previous chapter.

An example simulation of a tsunami system, simulated by the methods described above can be seen in figure 6.1.

6.2 A particle filter for the tsunami SPDE system

We again seek to approximate the filtering distribution, $p(X_{t_k}|Y_{1:k})$, and to do this we now adapt the Guided Proposal with Tempering and Mutation Filter (GPTMF) methodology that was introduced in Chapter 3. While the conceptual framework is the same, its application to the tsunami SPDE model introduces significant new challenges not present in the SDE case. The spatial discretisation of the SPDE leads to a very high-dimensional state space, and the driving noise is now a spatially-correlated, infinite-dimensional process rather than a vector of independent Brownian mo-



(a) $t = 25s$ (b) $t = 50s$ (c) $t = 75s$

(d) $t = 100s$ (e) $t = 125s$ (f) $t = 150s$

(g) $t = 175s$ (h) $t = 200s$ (i) $t = 225$

Figure 6.1: Example of a tsunami simulation driven by the SPDE (6.2). *Simulated on a regular 100 $\times$ 100 grid and with a time step of 0.1 seconds.*

tions.

6.2.1 Guided Proposals

As before, we seek to generate proposals that take the data into account and guide particles into regions of higher probability given the observed data point.

The process (6.2) for $t \in [t_{k-1}, t_k]$, conditioned on an observation y_k at time t_k and observed with Gaussian noise as defined in win (6.3), satisfies

$$dX_t = F(X_t)dt + \Sigma\nabla_x \log p(y_k|X_t)dt + \Sigma^{\frac{1}{2}}dW_t, \tag{6.5}$$

(see e.g. Rogers and Williams [2000]) where $p(y_k|X_t)$ denotes the likelihood of the observation at time t_k if the process evolved according to the unconditioned process (6.2) and ∇_x denotes the gradient with respect to the current state of the process.

We cannot simulate directly from (6.5) as the density $p(y_k|X_t)$ is not available analytically. A standard approach (see e.g. Schauer et al. [2017]) is to replace this likelihood with a tractable likelihood $\tilde{p}(y_k|X_t)$ that corresponds to the likelihood of the observation given that the underlying process evolves according to

$$dX_t = \Sigma^{\frac{1}{2}}dW_t, \tag{6.6}$$

i.e. a process that is spatially-correlated Brownian motion with covariance matrix Σ. It is then clear to see that $\tilde{p}(y_k|X_t)$ corresponds to the density of $Y = AX+\zeta$, where $\zeta \sim N(0, S)$ and $X \sim N(X_t, \Sigma(t_k-t))$. Standard results on multivariate normal distributions give that such a density is equal to the one of $N\left(AX_t, S+(t_k-t)A\Sigma A^T\right)$. Thus, we obtain

$$\log \tilde{p}(y_k|X_t) = -\tfrac{1}{2}(y_k - AX_t)^T(S+(t_k-t)A\Sigma A^T)^{-1}(y_k - AX_t) + \text{const}, \quad (6.7)$$

and therefore

$$\nabla_x \log \tilde{p}(y_k|X_t) = A^T(S+(t_k-t)A\Sigma A^T)^{-1}(y_k - AX_t). \quad (6.8)$$

The above derivations give us a proposal of the form

$$dZ_t = g(Z_t)dt + \Sigma^{\frac{1}{2}}dW_t, \quad (6.9)$$

where

$$g(Z_t) := F(Z_t) + \Sigma A^T(S+(t_k-t)A\Sigma A^T)^{-1}(y_k - AZ_t). \quad (6.10)$$

The proposal (6.9) is straightforward to simulate from and satisfies the conditions required to be absolutely continuous with respect to (6.2). Denote by $\mathbb{X}_{x,k}$ and $\mathbb{Z}_{x,k}$ the distributions on the space of continuous paths between time points t_{k-1} and t_k, for initial value $X_{t_{k-1}} = x$, induced by (6.2) and (6.9) respectively, then the Radon-Nikodym derivative of $\mathbb{X}_{x,k}$ with respect to $\mathbb{Z}_{x,k}$ is given by

$$\log \frac{d\mathbb{X}_{x,k}}{d\mathbb{Z}_{x,k}}(Z) = \int_{t_{k-1}}^{t_k} (F(Z_t) - g(Z_t))^T \Sigma^{-1} dW_t - \frac{1}{2} \int_{t_{k-1}}^{t_k} (F(Z_t) - g(Z_t))^T \Sigma^{-1} (F(Z_t) - g(Z_t)) dt. \tag{6.11}$$

The importance weight for a proposal path, Z_t, between observation times t_{k-1} and t_k is then proportional to

$$p(Y_k | Z_{t_k}) \frac{d\mathbb{X}_{x,k}}{d\mathbb{Z}_{x,k}}(Z_t). \tag{6.12}$$

The use of (6.9) as a proposal process should lead to proposed paths where the target density is higher and therefore particle weights that display less variance than if proposals were generated from the prior dynamics (6.2).

6.2.2 The tempering step

In the high-dimensional setting we find ourselves in, a data-informed proposal such as the one outlined in the previous section may not be enough to reduce the dissimilarity between target and proposal distributions to a sufficient level to combat the degeneracy in the particle weights, resulting in the estimate of the target distribution being dominated by a relatively small number of particles.

We attempt to improve this situation via the use of a sequence of intermediate distributions. This process is as outlined in section 3.2 and gives rise to a sequence of temperatures

$$0 = \phi_0 < \phi_1 < \cdots < \phi_M = 1 \tag{6.13}$$

and a corresponding sequence of intermediate path distributions. Denote by $\mathbb{X}_{x,k}$ and $\mathbb{Z}_{x,k}$ the distributions on the space of continuous paths between time points t_{k-1} and t_k, for initial value $X_{t_{k-1}} = x$, induced by (6.2) and (6.9) respectively, then this sequence of distributions is

$$\begin{gathered}
\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{X}_{x,k}}{d\mathbb{Z}_{x,k}}(X))^{\phi_0}, \\
\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{X}_{x,k}}{d\mathbb{Z}_{x,k}}(X))^{\phi_1}, \\
\cdots \\
\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{X}_{x,k}}{d\mathbb{Z}_{x,k}}(X))^{\phi_M}.
\end{gathered} \tag{6.14}$$

These intermediate distributions aim to bridge the gap between the proposal and target distributions. In particular, we perform iterative importance sampling to go from the proposal to the target distribution. Note that $\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{X}}{d\mathbb{Z}}(X))^{\phi_0} = \mathbb{Z}_{x,k}(dX)$ and $\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{X}}{d\mathbb{Z}}(X))^{\phi_M} = \mathbb{X}_{x,k}(dX|Y_k)$.

It remains to be determined how many intermediate distributions to use and how to choose the temperatures. We propose to choose the temperatures on the fly such that the effective sample size of the particles remains above a certain threshold. If we are at the kth time step of the algorithm and have so far executed $j-1$ tempering steps, the particle weights will be equal across all particles due to resampling at the end of the previous tempering step. The weight of the ith particle at the jth tempering step, with proposal path Z_t^i for $t_{k-1} < t \leqslant t_k$ starting at $Z_{t_{k-1}}^i = x_{k-1}^i$ can be given as a function of the next temperature

$$W_{k,j}^i(\phi) \propto \left(\frac{d\mathbb{X}_{x_{k-1}^i,k}}{d\mathbb{Z}_{x_{k-1}^i,k}}(Z_t^i) p(Y_k | Z_{t_k}) \right)^{\phi - \phi_{j-1}}. \tag{6.15}$$

The effective sample size of the particles can then be expressed as a function of ϕ as

$$ESS(\phi) := \frac{1}{\sum_{i=1}^N (W_{k,j}^i(\phi))^2}. \tag{6.16}$$

The next temperature is then as the solution to the equation $ESS(\phi) = ESS_{min}$, where ESS_{min} is a user-defined minimum effective sample size. This is straightforward to solve numerically, e.g. using a bisection method.

6.2.3 The mutation step

Resampling multiple times in the tempering step outlined above can lead to a lack of sample diversity in the particles. To combat this we use a small (user-defined) number of steps from an MCMC procedure to increase the diversity of the samples.

More specifically, at the kth time step and jth temperature in our tempering procedure, we wish to specify a Markov kernel, $K_{k,l}(dX', X)$ that preserves the intermediate distribution $\mathbb{Z}_{x,k}(dX)(p(Y_k|X_{t_k})\frac{d\mathbb{Z}_{x,k}}{d\mathbb{X}_{x,k}}(X))^{\phi_j}$.

Following the approach in section 3.3.2 and again exploiting the fact that a proposal of the form (6.9) is uniquely determined by the driving noise, we can follow a pCN scheme.

We first generate a new sample for the noise as

$$W' = \rho W + \sqrt{1-\rho^2}\xi \tag{6.17}$$

where ρ is a user-defined parameter controlling the step size, W is the initial driving noise and ξ is a spatially-correlated Brownian motion with increments drawn from the Gaussian random field with covariance matrix Σ.

We then use this new noise sample to generate a new proposal, Z', and accept this new proposal with acceptance probability

$$\alpha(Z, Z') = 1 \wedge \frac{(p(Y_k|Z'_{t_k})\frac{d\mathbb{Z}_{x,k}}{d\mathbb{X}_{x,k}}(Z'))^{\phi_j}}{(p(Y_k|Z_{t_k})\frac{d\mathbb{Z}_{x,k}}{d\mathbb{X}_{x,k}}(Z))^{\phi_j}}. \tag{6.18}$$

The above scheme is straightforward to implement as increments for the initial driving noise can be stored in memory and new increments can be quickly generated via FFT using the methods outlined in section 5.3.2.

6.3 Computational considerations

The GPTMF is a computationally intensive algorithm, requiring many simulations and weight calculations per particle and pCN step, but using the methods outlined in Chapter 5 can help to improve the efficiency.

For simulating paths of the proposal process (6.9) and for generating new proposal paths in the pCN step (6.17) we are required to simulate samples from a Gaussian random field. We can do this efficiently via FFT as explained in section 5.3.2.

Both the proposal process (6.9) and the calculation of weights (6.11) require a matrix vector multiplication where the matrix is either the covariance matrix Σ or its inverse. The method described in section 5.3.1 can be used to carry out these multiplications efficiently, again using the FFT.

In addition to the efficiencies provided by using the FFT, further speed enhancements can be achieved through the use of so called "FFT plans." An FFT plan, available in many popular FFT libraries (for example, we use FFTW [Frigo and Johnson, 2005] in this thesis), contains all the necessary information to perform an FFT (or inverse FFT) operation on a given input size. This plan sets up an optimized computational strategy to execute the operation as quickly as possible. In our case, the input size is determined by the spatial resolution and known at runtime, allowing us to compute the FFT plan once and reuse it throughout the GPTMF algorithm.

The GPTMF is well suited for parallel computation. Both the proposal step and the mutation step operate independently across particles and are computationally intensive enough that their execution time outweighs any communication overhead between processes. As a result, parallel computation can lead to significant efficiency gains.

6.4 Numerical results

The numerical experiments in this section are designed with a physical setup that is broadly similar to the one used in Chapter 5. However, the fundamental difference lies in the underlying dynamics: the model is now the full SPDE (6.1), where noise is integrated continuously, rather than at discrete intervals of a deterministic PDE. The specific parameters for this experiment are as follows:

- 2D-domain $\Omega = [0, P] \times [0, Q]$ with $P = Q = 40km$, discretised as in 5.2, with $p = q = 50$.
- Constant sea depth of $D(x, y) = 1500m$.
- Boundary damping factor $\lambda = 0.015$, with a boundary size of 5 grid points.
- Independent observation noise across stations, distributed $N(0, c^2)$, with $c = 0.0025m$.
- Model noise independent across height and horizontal and vertical velocity fields, with covariance matrices $\Sigma_1, \Sigma_2, \Sigma_3$ respectively. These covariance matrices correspond to the covariance function

$$r(x, y) = \sigma^2 \exp\left(-\sqrt{\left(\frac{x^2 + y^2}{l^2}\right)}\right). \tag{6.19}$$

  where we set the correlation length $l = 3km$ for all three fields, $\sigma = 0.01m$ for the covariance of the height field Σ_1 and $\sigma = 1$ for the covariance of the velocity fields, Σ_2, Σ_3.
- Initialise the wave at the point $(6500m, 6500m)$, with an initial spread $s = 3000m$ and initial max height, $h_{\text{max}} = 3m$.
- $N = 500$ particles, with an ESS threshold of 250 in the tempering

steps.

- Simulate for a total of 220 seconds, with measurements taken every second from 70 seconds onwards.
- Observations taken from 25 observation stations, spaced in a regular grid at the centre of our domain. Side length of square approximately 12km.
- Simulations of SDEs done with a time step of 0.05 seconds.
- Mutation steps are carried out by synthesising 5 pCN steps with $\rho = 0.9$.

We compare the performance of our algorithm to a bootstrap filter, where the proposal is generated from the prior dynamics (6.2) and no tempering or mutation steps are carried out.

Figure 6.2 shows the posterior mean of the wave height field estimated by both the bootstrap filter and GPTMF at four different time points, along with the true wave height field at these time points. It is clear to see that the bootstrap filter struggles to estimate the true signal in comparison to GPTMF. The GPTMF is able to detect the wave when it is observed by the first few observation stations and track it as it moves across the domain.

Figure 6.3 shows the actual height and residuals of the height at four different stations across all time points. The residuals are calculated as the difference between the true height field and the posterior mean of the height field estimated by the GPTMF. These residuals appear are very small across all stations and time points, indicating that the GPTMF is able to estimate the true signal well. The 95 percent confidence are also plotted and for the majority of time points zero is within the confidence interval, indicating that the value of the true signal is within the 95 percent confidence interval of the posterior mean.

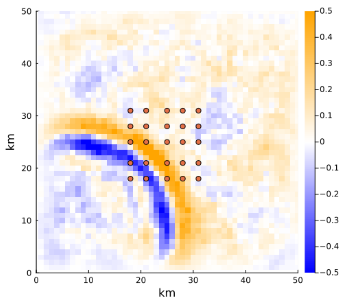
Figure 6.4 (along with figures in Appendix B) show the estimated 1-d probability density functions of the wave height at four different time points at the locations of the observation stations. The true height value and the observed height value are also plotted for comparison. These distributions naturally tend to be centred at the measured height but in the majority of cases seem to possess sufficient spread so that the actual heights falls in an area of high probability density, an indication that the GPTMF is able to estimate the filtering density well.

We also carry out a second numerical experiment, conditions are the same as stated previously except for the following changes:

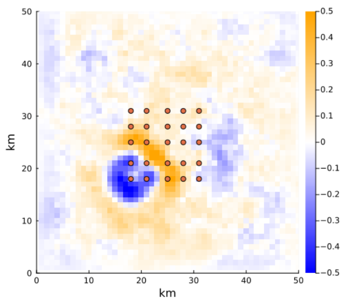
- The number of particles is reduced to $N = 200$, and ESS threshold is reduced to 100.
- Run the filter for 40 observations from time 80s to time 120s.
- We run for multiple different values of p and q to see how the algorithm performs with different spatial resolutions.
- Run for different values of c to see how the algorithm performs with different levels of observation noise.
- Run for different numbers of observation stations, again arranged in a regular grid at the centre of the domain.

- Run each experiment 10 times.

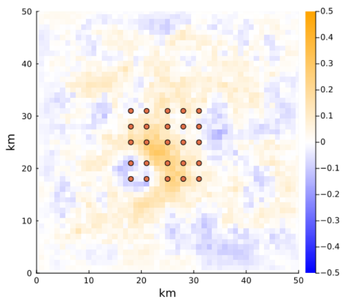
Results from these experiments can be found in table 6.1. As expected, we see that the required number of temperatures increases as we make the observations more informative, either through a reduction in the observation noise or an increase in the number of observation stations. We also see that the algorithm seems robust to changes in spatial resolution, with the number of temperatures required remaining relatively similar for different values of p and q. Though the number of temperatures is fairly robust to changes in the spatial resolution we can see that increasing the spatial resolution significantly increases the runtime of the algorithm, while increasing the spatial resolution should increase the accuracy of the algorithm it can incur a heavy cost in terms of computation, this represents an important trade-off for practical applications. Experiments were run on a GCP c2d-standard-8 instance, with 8vCPUs and 32GB RAM; significant speed improvements could be achieved could be achieved by using a machine with more CPUs to take full advantage of the embarrassingly parallel nature of the proposal and mutation steps across particles.



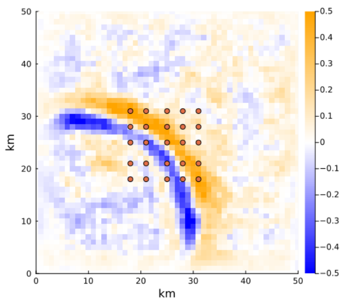
(a) $t = 130s$, true signal



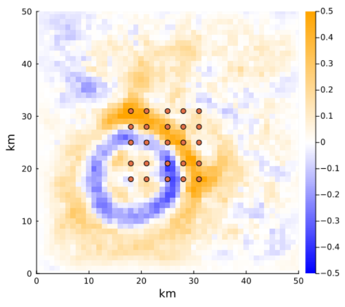
(b) $t = 130s$, GPTMF



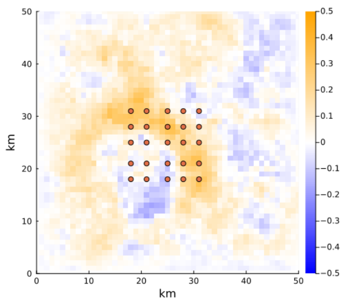
(c) $t = 130s$, bootstrap



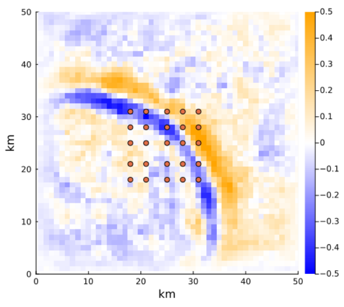
(d) $t = 160s$, true signal



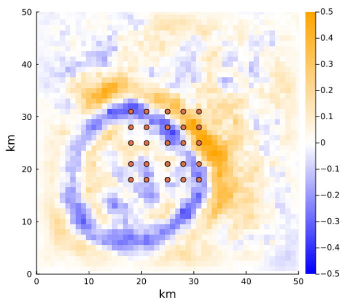
(e) $t = 160s$, GPTMF



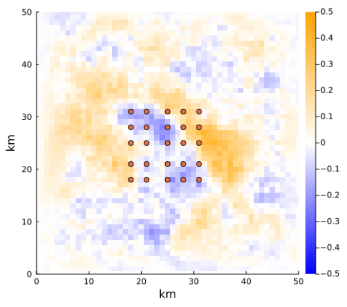
(f) $t = 160s$, bootstrap



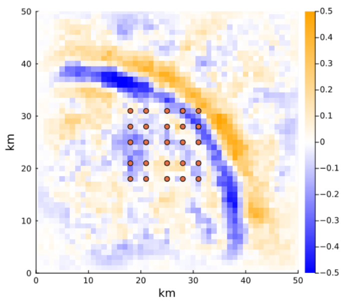
(g) $t = 190s$, true signal



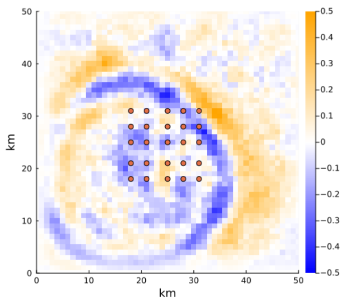
(h) $t = 190s$, GPTMF



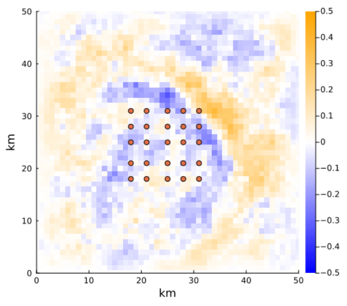
(i) $t = 190s$, bootstrap



(j) $t = 220s$, true signal



(k) $t = 220s$, GPTMF



(l) $t = 220s$, bootstrap

Figure 6.2: Estimates of wave height via posterior means of GPTMF and bootstrap particle filters, compared with actual wave heights.

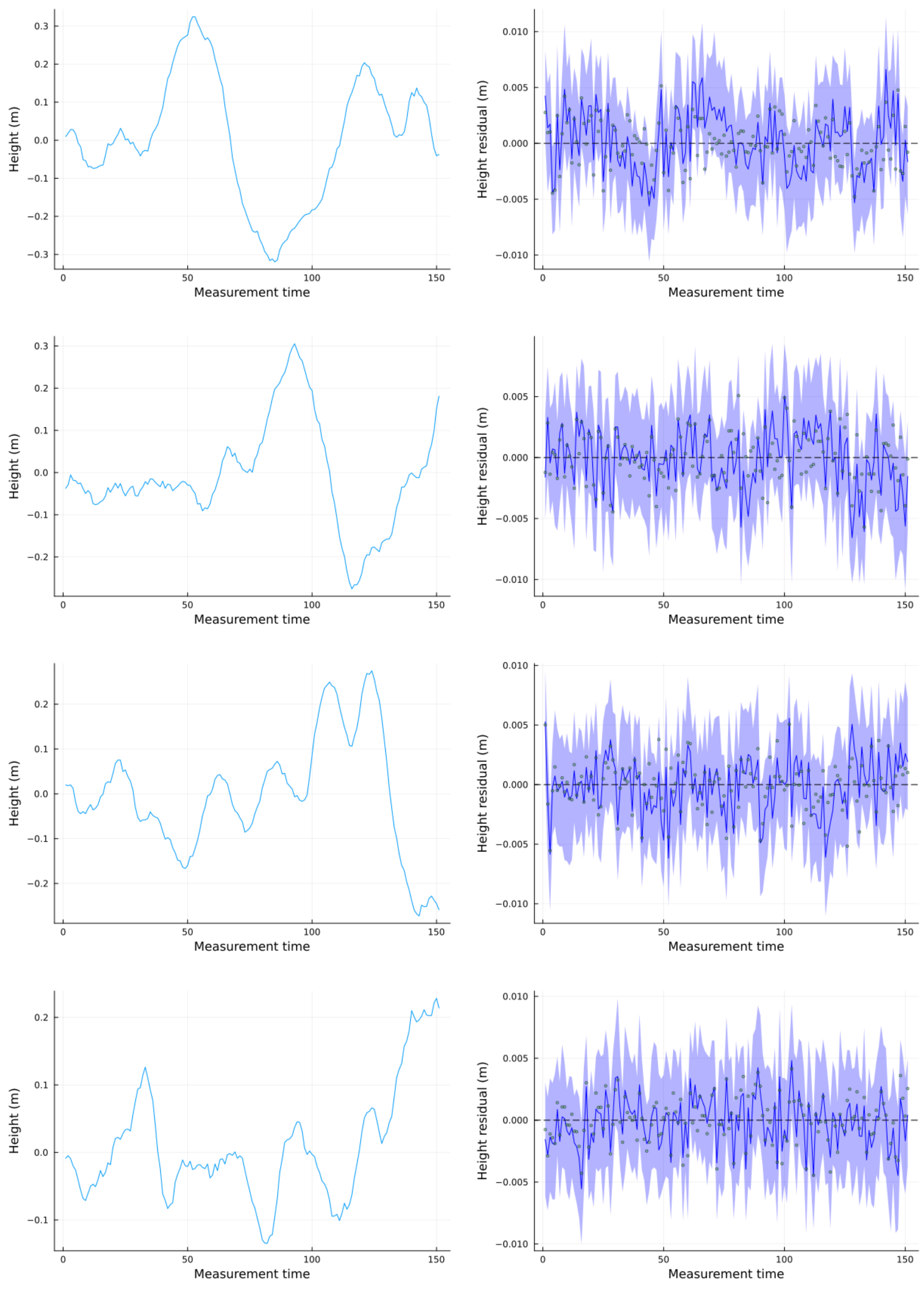


Figure 6.3: Plot of actual heights and the residual of the height estimates at four different stations across time. Shaded blue area corresponds to the 95% confidence intervals and the green dots are the residual value of the observations relative to the actual height. Plots are for station positions (16.8km, 16.8km), (20km, 20km), (22.4km, 22.4km), (24.8km, 24.8km).

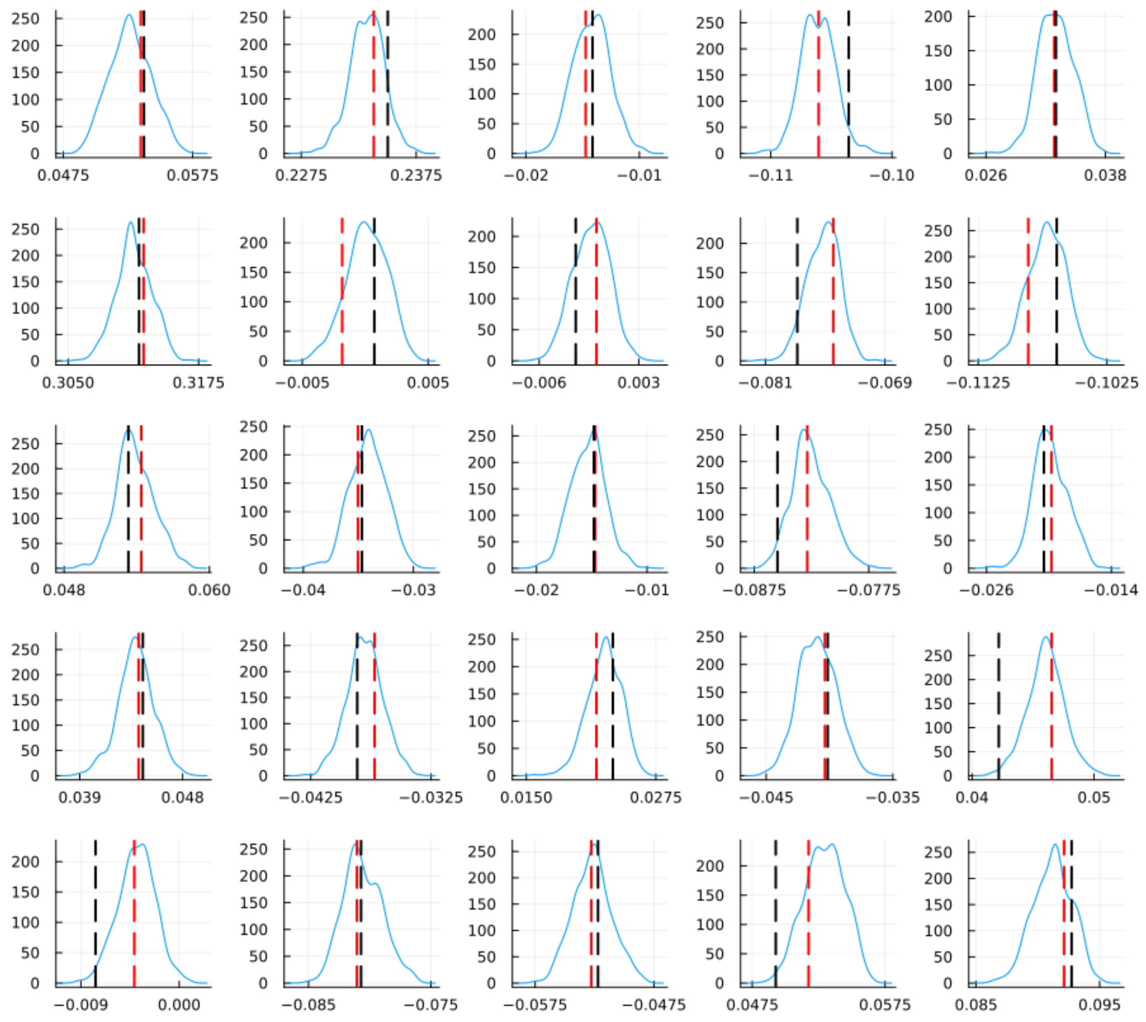


Figure 6.4: Estimated probability density functions of wave height via GPTMF at all station locations at 130 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

c	Grid square size	No. stations	No. temp	ESS	Runtime (s)
0.0025	50 $\times$ 50	16	3.23	112	336
0.0025	50 $\times$ 50	25	4.89	110	426
0.0025	50 $\times$ 50	36	6.36	107	597
0.0025	75 $\times$ 75	16	3.39	113	3027
0.0025	75 $\times$ 75	25	5.07	108	4915
0.0025	75 $\times$ 75	36	7.12	107	5822
0.0025	100 $\times$ 100	16	3.95	111	13297
0.0025	100 $\times$ 100	25	5.31	103	14870
0.0025	100 $\times$ 100	36	6.54	109	19152
0.005	50 $\times$ 50	16	2.85	114	305
0.005	50 $\times$ 50	25	3.91	109	413
0.005	50 $\times$ 50	36	5.91	108	569
0.01	50 $\times$ 50	16	2.60	114	281
0.01	50 $\times$ 50	25	3.64	110	374
0.01	50 $\times$ 50	36	5.36	108	495

Table 6.1: Average number of temperatures per observation and the average effective sample size when running the GPTMF over different values of observation noise, spatial resolution and number of observation stations. Results are based on average of ten independent runs for each set of values.

Chapter 7

Conclusions and Further Work

The main contributions of this thesis can be summarised as follows:

- **Development of a Sequential Monte Carlo methodology for partially observed SDEs**: This method approximates the filtering distribution of stochastic differential equations observed at discrete time steps with added observation noise. The approach integrates guided proposals, tempering steps that target intermediate distributions between the prior and posterior densities, and mutation steps executed through MCMC methods.
- **Efficient implementation of an optimal filter for tsunami wave field data assimilation**: This contribution addresses data assimilation when the problem is modelled as a discrete-time state-space model. By leveraging existing results that utilize the Fast Fourier Transform to efficiently sample from Gaussian random fields, we adapt and apply these techniques within a filtering framework.
- **Extension of the filtering methodology from SDEs to SPDEs**: This extension tackles the complexities introduced by spatial correlations in the driving noise, presenting a challenging scenario. Our focus remains on the data assimilation of tsunami wave fields, demonstrating the robustness of the algorithm through numerical experiments on

synthetic data.

There are many ways this work could be extended, some examples are:

- **Improved proposals**: The methodology described in Chapters 3 and 6 could be further refined by improving any of its components, such as the proposal, tempering, or mutation steps. Specifically, a proposal that more accurately targets the filtering distribution would lead to an immediate improvement in the algorithm's performance by reducing the variance of particle weights immediately following the proposal step. This reduction would, in turn, decrease the number of temperatures required and enhance particle diversity. Recent work demonstrating the success of improved proposals includes Chopin et al. [2023], where neural networks are used to generate improved proposals by approximating transformations of stochastic processes that are typically intractable.

- **Application to Real-World data:** So far we have only applied the proposed methodology to synthetically generated tsunami height data. Testing on real-world data is a natural next step and would further assess the robustness, accuracy, and practical applicability of the methodology in predicting and modelling tsunami behaviour. Real-world data provides additional complexities when compared to synthetic data and may lead to limitations being discovered and further refinements to be made to the methodology. Historical tsunami data could be obtained from the Deep-ocean Assessment and Reporting of Tsunamis (DART) system [Gonzalez et al., 1998].

- **Optimal filter on irregular grids:** One of the key assumptions in developing the efficient implementation of the optimal filter in chapter 5 is that the space is discretised into a regular grid - this allowed us to use the methods of Dietrich and Newsam [1996] to efficiently generate realisations of a Gaussian random field. We could extend this work to irregular grids, by making use of the integrated nested Laplace approximation (INLA) approach (see e.g.[Krainski et al., 2018]).

- **Localised particle filters:** Localised particle filters have been proposed as a method to address particle degeneracy when filtering in contexts with spatial dependence. By exploiting the weak dependence between distant points, these methods enhance the performance of particle filters and mitigate particle degeneracy. For an example of such an approach, refer to Graham and Thiery [2019]. We could explore integrating this localisation strategy into the methodology discussed in Chapter 6.

Bibliography

Christophe Andrieu, Gareth O Roberts, et al. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

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

Alexandros Beskos, Omiros Papaspiliopoulos, Gareth O Roberts, and Paul Fearnhead. Exact and computationally efficient likelihood-based estimation for discretely observed diffusion processes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):333–382, 2006.

Alexandros Beskos, Omiros Papaspiliopoulos, and Gareth O Roberts. A factorisation of diffusion measure and finite sample path constructions. *Methodology and Computing in Applied Probability*, 10(1):85–104, 2008a.

Alexandros Beskos, Gareth Roberts, Andrew Stuart, and Jochen Voss. MCMC methods for diffusion bridges. *Stochastics and Dynamics*, 8(03): 319–350, 2008b.

Alexandros Beskos, Konstantinos Kalogeropoulos, and Erik Pazos. Advanced MCMC methods for sampling on diffusion pathspace. *Stochastic Processes and their Applications*, 123(4):1415–1453, 2013.

James Carpenter, Peter Clifford, and Paul Fearnhead. Improved particle filter for nonlinear problems. *IEE Proceedings-Radar, Sonar and Navigation*, 146(1):2–7, 1999.

Charles Cerjan, Dan Kosloff, Ronnie Kosloff, and Moshe Reshef. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50(4):705–708, 1985.

Nicolas Chopin, Andras Fulop, Jeremy Heng, and Alexandre H Thiery. Computational Doob h-transforms for online filtering of discretely observed diffusions. In *International Conference on Machine Learning*, pages 5904–5923. PMLR, 2023.

Nicolas Chopin et al. Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.

William Coffey and Yu P Kalmykov. *The Langevin equation: with applications to stochastic problems in physics, chemistry and electrical engineering*, volume 27. World Scientific, 2012.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90): 297–301, 1965.

Simon L Cotter, Gareth O Roberts, Andrew M Stuart, and David White. MCMC methods for functions: modifying old algorithms to make them faster. *Statistical Science*, pages 424–446, 2013.

Philip J Davis. *Circulant matrices*. American Mathematical Soc., 2013.

Pierre Del Moral. Feynman-Kac formulae, Probability and its Applications, 2004.

Pierre Del Moral and Lawrence M Murray. Sequential Monte Carlo with highly informative observations. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):969–997, 2015.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

Bernard Delyon and Ying Hu. Simulation of conditioned diffusion and application to parameter estimation. *Stochastic Processes and their Applications*, 116(11):1660–1675, 2006.

Colin R Dietrich and Garry N Newsam. A fast and exact method for multidimensional Gaussian stochastic simulations. *Water Resources Research*, 29(8):2861–2869, 1993.

CR Dietrich and GN Newsam. A fast and exact method for multidimensional Gaussian stochastic simulations: Extension to realizations conditioned on direct and indirect measurements. *Water Resources Research*, 32(6): 1643–1652, 1996.

R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69, 2005.

Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12(656-704):3, 2009.

Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

Garland B Durham and A Ronald Gallant. Numerical techniques for maximum likelihood estimation of continuous-time diffusion processes. *Journal of Business & Economic Statistics*, 20(3):297–338, 2002.

Morris L Eaton. *Multivariate statistics: A vector space approach*. John Wiley & Sons, 1983.

Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

Paul Fearnhead, Omiros Papaspiliopoulos, and Gareth O Roberts. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):755–777, 2008.

Matteo Frigo and Steven G Johnson. The design and implementation of

FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

Mathieu Gerber, Nicolas Chopin, Nick Whiteley, et al. Negative association, ordering and convergence of resampling methods. *The Annals of Statistics*, 47(4):2236–2260, 2019.

John Geweke. Bayesian Inference in Econometric Models Using Monte Carlo Integration. *Econometrica*, 57:1317–39, 1989.

Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. *Markov chain Monte Carlo in practice*, volume 2. Chapman & Hall, 1996.

Narendra S Goel and Nira Richter-Dyn. *Stochastic models in biology*. Elsevier, 2016.

Andrew Golightly and Darren J Wilkinson. Bayesian inference for nonlinear multivariate diffusion models observed with error. *Computational Statistics & Data Analysis*, 52(3):1674–1693, 2008.

Frank I Gonzalez, Hank M Milburn, Eddie N Bernard, and Jean C Newman. Deep-ocean assessment and reporting of tsunamis (DART): Brief overview and status report. In *Proceedings of the international workshop on Tsunami Disaster Mitigation*, volume 19, page 2. NOAA Tokyo, Japan, 1998.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

Matthew M Graham and Alexandre H Thiery. A scalable optimal-transport based local particle filter. *arXiv preprint arXiv:1906.00507*, 2019.

Robert M Gray. Toeplitz and circulant matrices: A review. 2006.

W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. 1970.

Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007.

Masaaki Kijima. *Stochastic processes with applications to finance*. Chapman and Hall/CRC, 2002.

Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

Elias Krainski, Virgilio Gómez-Rubio, Haakon Bakka, Amanda Lenzi, Daniela Castro-Camilo, Daniel Simpson, Finn Lindgren, and Håvard Rue. *Advanced spatial modeling with stochastic partial differential equations using R and INLA*. Chapman and Hall/CRC, 2018.

François LeGland and Laurent Mevel. Recursive estimation in hidden Markov models. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 4, pages 3468–3473. IEEE, 1997.

Jun S Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.

Takuto Maeda, Kazushige Obara, Masanao Shinohara, Toshihiko

Kanazawa, and Kenji Uehira. Successive estimation of a tsunami wavefield without earthquake source data: A data assimilation approach toward real-time tsunami forecasting. *Geophysical Research Letters*, 42(19): 7923–7932, 2015.

Terry A Marsh and Eric R Rosenfeld. Stochastic processes for interest rates and equilibrium bond prices. *The Journal of Finance*, 38(2):635–646, 1983.

Gisiro Maruyama. Continuous Markov processes and stochastic equations. *Rendiconti del Circolo Matematico di Palermo*, 4(1):48–90, 1955.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

Thomas Mikosch. *Elementary stochastic calculus, with finance in view*, volume 6. World Scientific Publishing Company, 1998.

Radford M Neal et al. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

Bernt Øksendal. *Stochastic differential equations: An introduction with applications*. Springer, 5 edition, 1998.

Omiros Papaspiliopoulos. Monte Carlo probabilistic inference for diffusion processes: A methodological framework. *Bayesian Time Series Models*, pages 82–103, 2011.

Michael K Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

George Poyiadjis, Arnaud Doucet, and Sumeetpal S Singh. Particle approximations of the score and observed information matrix in state space models with application to parameter estimation. *Biometrika*, 98(1):65–80, 2011.

Gareth O Roberts, Richard L Tweedie, et al. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2 (4):341–363, 1996.

L Chris G Rogers and David Williams. *Diffusions, Markov processes, and martingales: Itô calculus*, volume 2. Cambridge University Press, 2000.

Tomasz Rolski, Hanspeter Schmidli, Volker Schmidt, and Jozef L Teugels. *Stochastic processes for insurance and finance*, volume 505. John Wiley & Sons, 2009.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Simo Särkkä, Tommi Sottinen, et al. Application of Girsanov theorem to particle filtering of discretely observed continuous-time non-linear systems. *Bayesian Analysis*, 3(3):555–584, 2008.

Moritz Schauer, Frank Van Der Meulen, Harry Van Zanten, et al. Guided proposals for simulating multi-dimensional diffusion bridges. *Bernoulli*, 23(4A):2917–2950, 2017.

Kazimierz Sobczyk. *Stochastic differential equations: with applications to physics and engineering*, volume 40. Springer Science & Business Media, 2013.

Harold Wayne Sorenson. *Kalman filtering: theory and application*. IEEE, 1985.

Tamas Szabados. An elementary introduction to the Wiener process and stochastic integrals. *Studia Scientiarum Mathematicarum Hungarica*, 31 (arXiv: 1008.1510):249–297, 2010.

Luke Tierney. A note on Metropolis-Hastings kernels for general state spaces. *Annals of Applied Probability*, pages 1–9, 1998.

Henry C Tuckwell. *Stochastic processes in the neurosciences*, volume 56. SIAM, 1989.

Geoffrey K Vallis. *Atmospheric and oceanic fluid dynamics*. Cambridge University Press, 2017.

Nicolaas Godfried Van Kampen. *Stochastic processes in physics and chemistry*, volume 1. Elsevier, 1992.

Charles Van Loan. *Computational frameworks for the fast Fourier transform*. SIAM, 1992.

Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002.

Wolfgang Wagner. Monte Carlo evaluation of functionals of solutions of stochastic differential equations. Variance reduction and numerical examples. *Stochastic Analysis and Applications*, 6(4):447–468, 1988.

Gavin A Whitaker, Andrew Golightly, Richard J Boys, and Chris Sherlock. Improved bridge constructs for stochastic differential equations. *Statistics and Computing*, 27(4):885–900, 2017.

Nick Whiteley and Adam M Johansen. Recent developments in auxiliary particle filtering. *Inference and Learning in Dynamic Models*, 38:39–47, 2010.

Darren J Wilkinson. *Stochastic modelling for systems biology*. Chapman and Hall/CRC, 2006.

Eugene Wong and Bruce Hajek. *Stochastic processes in engineering systems*. Springer Science & Business Media, 2012.

Appendix A

Plots for optimal filter

This appendix presents probability density functions of wave height estimated using the optimal filter methodology introduced in Chapter 5. These plots show the filtering distributions at various time points and observation station locations, with the black dashed lines indicating actual wave heights and red dashed lines showing the observed measurements.

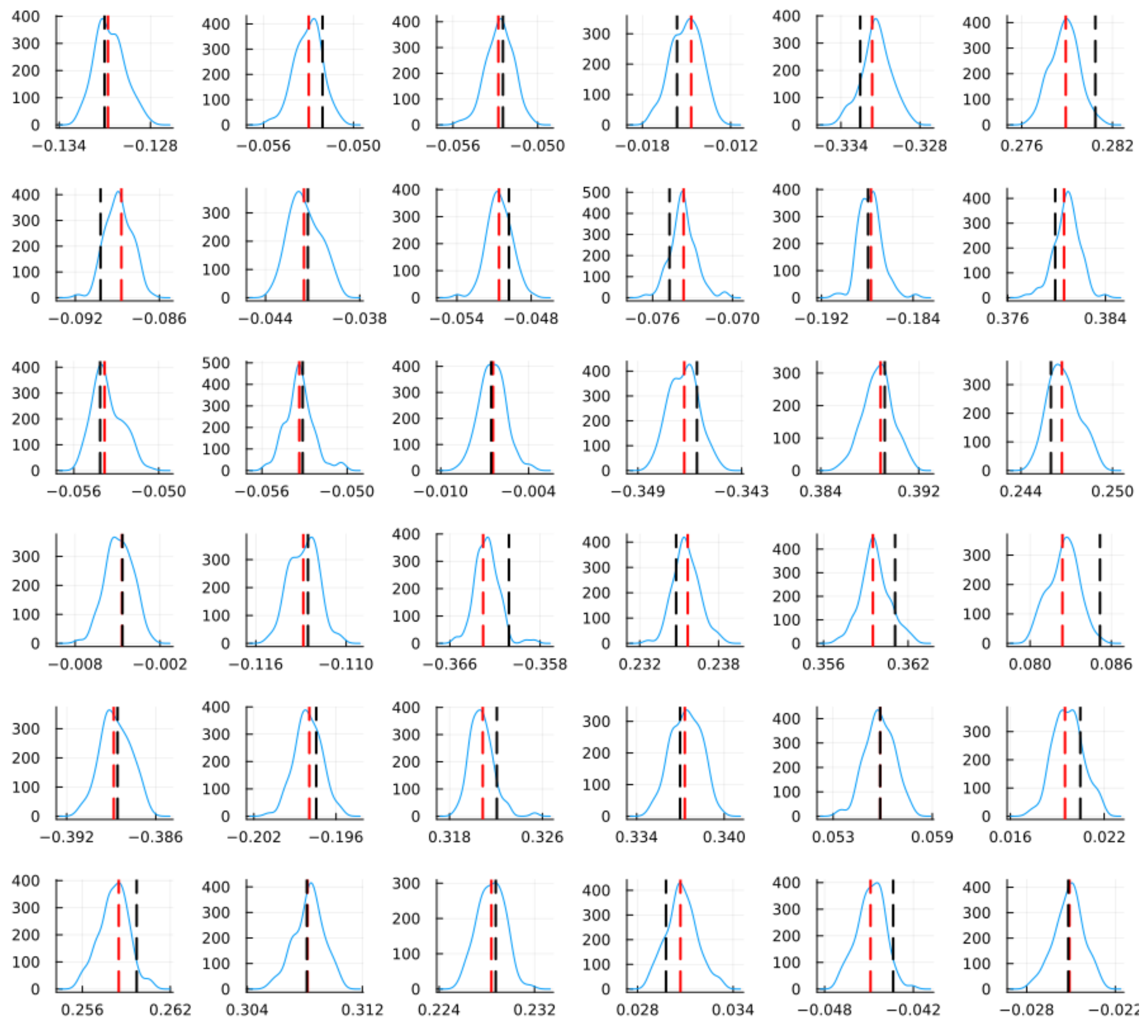


Figure A.1: Estimated probability density functions of wave height via the optimal filter at all station locations at 170 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

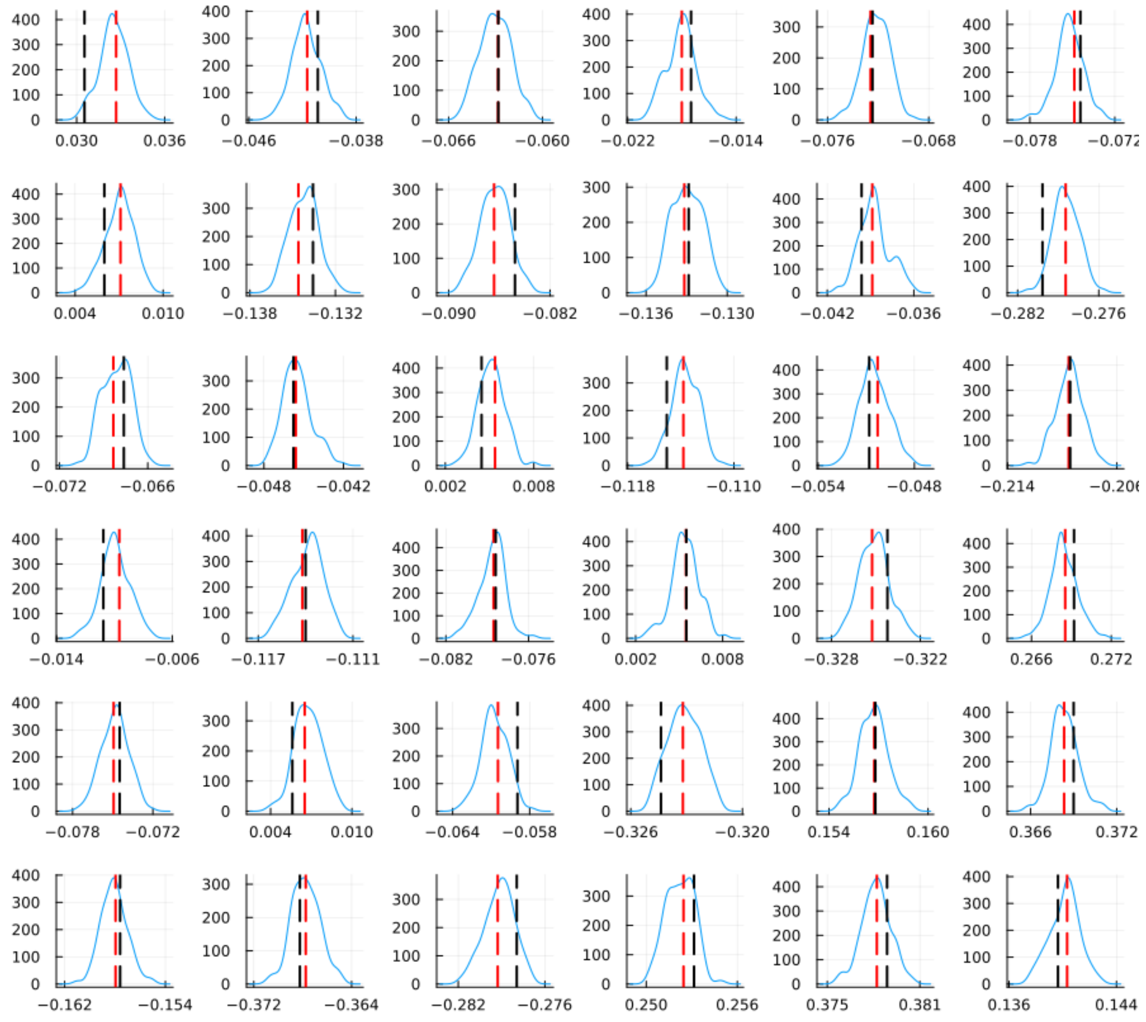


Figure A.2: Estimated probability density functions of wave height via the optimal filter at all station locations at 200 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

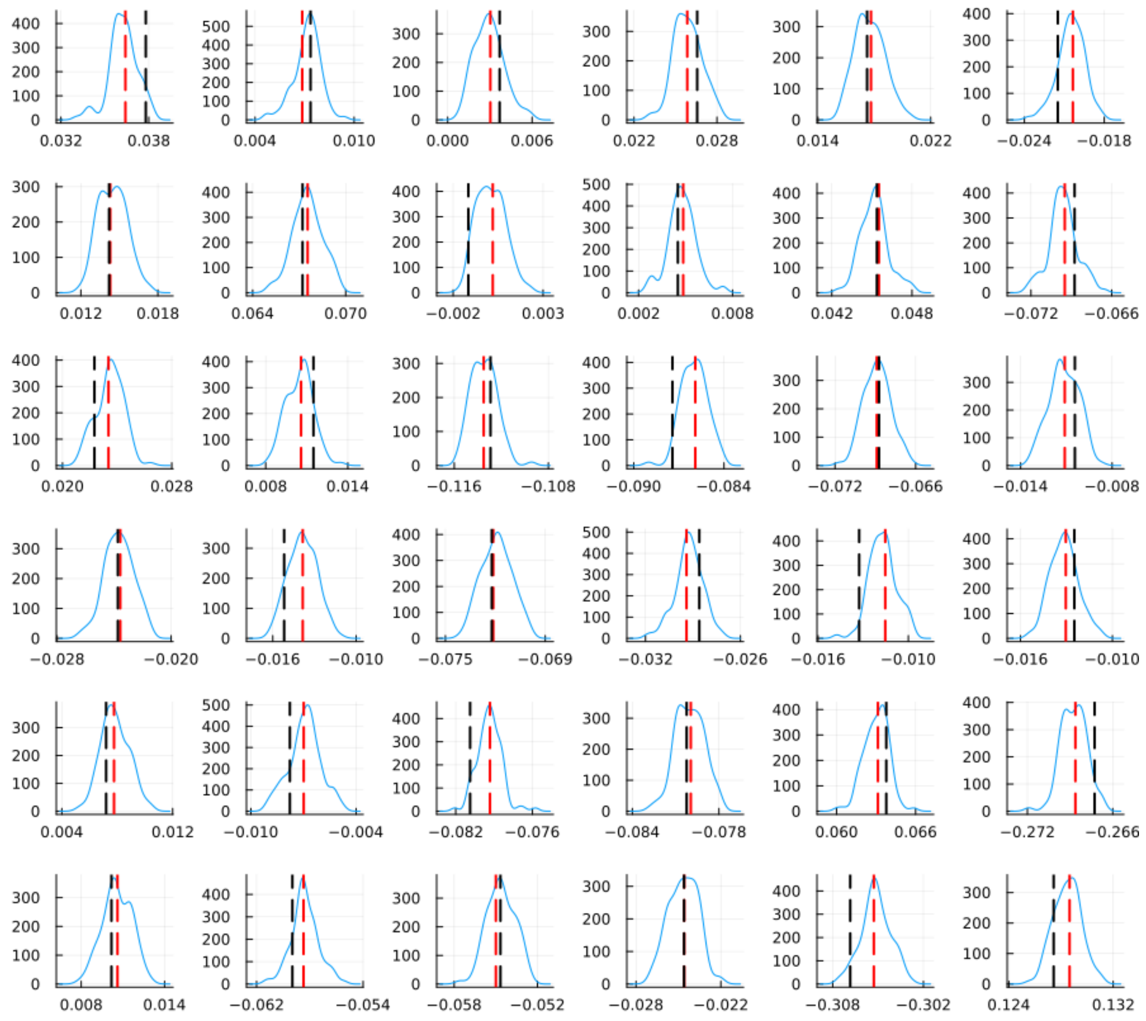
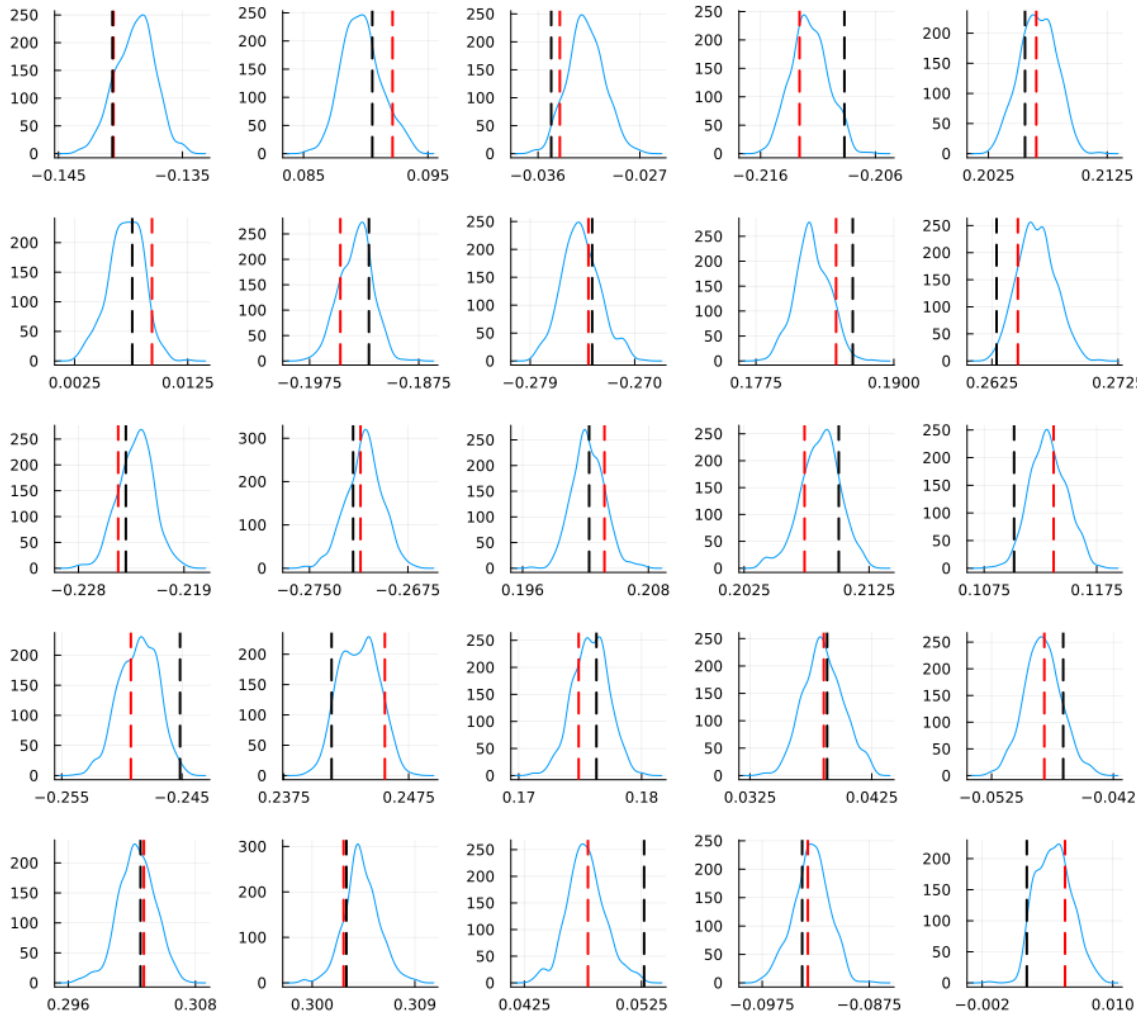


Figure A.3: Estimated probability density functions of wave height via the optimal filter at all station locations at 230 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

Appendix B

Plots for GPTMF

This appendix contains probability density functions of wave height estimated using the GPTMF methodology, applied to the tsunami SPDE model in Chapter 6. The plots show filtering distributions at various time points and observation station locations, with black dashed lines indicating actual wave heights and red dashed lines showing the observed measurements.



(a)

Figure B.1: Estimated probability density functions of wave height via GPTMF at all station locations at 160 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.

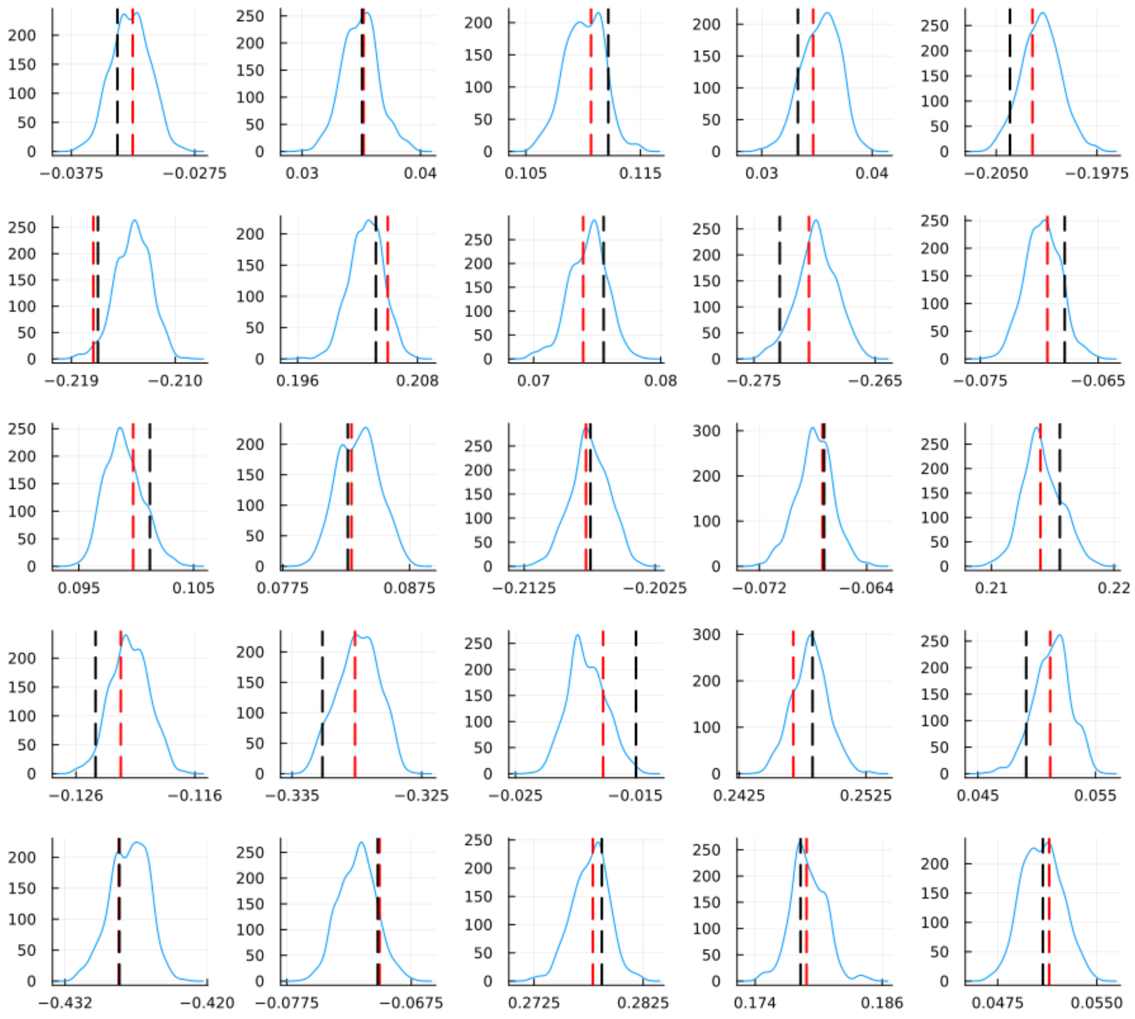
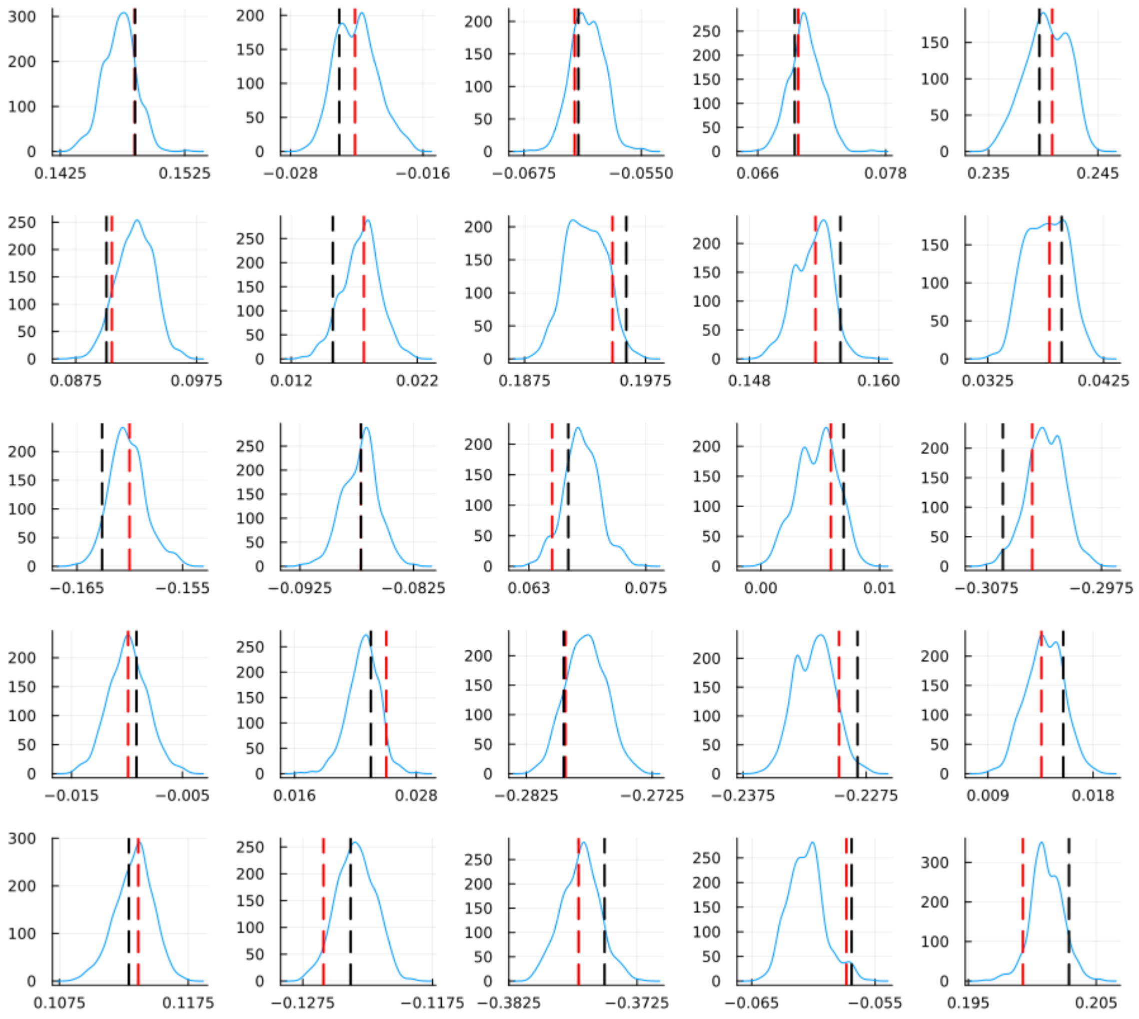

(a)

Figure B.2: Estimated probability density functions of wave height via GPTMF at all station locations at 190 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.



(a)

Figure B.3: Estimated probability density functions of wave height via GPTMF at all station locations at 220 seconds. The black dashed line shows the actual wave height at the stations and the dashed red line shows the observed wave height.