Disease Surveillance using Bayesian Methods

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor in Philosophy by

Conor Rosato

October 2023

Abstract

Developing Markov Chain Monte Carlo (MCMC) algorithms has been an active area of research. Extensions of the original Metropolis-Hastings random walk (MHRW) algorithm, such as Metropolis-adjusted Langevin algorithm (MALA), Hamiltonian Monte Carlo (HMC) and the No-U-Turn Sampler (NUTS), include gradient information about the posterior when proposing parameters in areas of higher probability within the target. Particle-Markov Chain Monte Carlo (p-MCMC) is a similar parameter estimation algorithm that utilises a particle filter to calculate an unbiased estimate of the log-likelihood which can be used in the MHRW algorithm. However, as noted in the literature, obtaining gradients of the log-likelihood w.r.t the parameters is difficult due to operations inherent to the particle filter being non-differentiable. This obstacle has hindered the use of gradient based proposals within p-MCMC. Therefore, in this thesis, a novel method for obtaining the gradient of the log-likelihood w.r.t the parameters by fixing the random number seed within the particle filter is considered. This allows the particle filter to be posed as a deterministic function, i.e. running the particle filter multiple times will result in the same resampling realisations, log-likelihood and associated gradient estimates. When a different resampling realisation occurs between two parameter values, a piecewise continuous estimate of the log-likelihood and gradient occurs.

It is shown that these estimates are still compatible with gradient based proposals such as MALA, HMC and NUTS. A comparison of these samplers is made when estimating the parameters of two state-space models. Results indicate that although NUTS can make multiple gradient evaluations per MCMC iteration, it can produce more accurate estimates in shorter computation time. Frameworks for describing the differentiable particle filter and NUTS in PyTorch and PyMC3, respectively are also provided. This allows the derivatives and partial derivatives to be calculated via automatic differentiation. Particle filters have been used extensively to model and track infectious disease epidemics, with p-MCMC used to estimate the parameters of these models. Although gradient based proposals are used in non-particle methods when modelling epidemiology, the standard proposal when using p-MCMC is the MHRW. Applying the novel differentiable particle filter to two epidemiological models, NUTS can recover the correct parameters in shorter run time when compared to the MHRW proposal.

In the context of epidemiological modelling it is essential for public health officials to

understand how a disease spreads through a population. This has recently come to the forefront with the emergence of COVID-19. At the beginning of the pandemic it was vital to gather accurate open-source datasets from which to infer how quickly the virus was spreading. As well as parameter estimation, MCMC algorithms have the ability to make forecasts of quantities of interest. Evaluating these predictions with simple scoring rules gives an indication of how well the model represents reality. The scoring rule normalised estimation error squared (NEES) can detect shortcomings within a model such as incorrect parameters, resulting in forecasts that are over-confident or over-cautious. A detailed description of why being cautious rather than confident is more desirable is provided.

NEES can also be used when evaluating the effectiveness of different open-source datasets when making future predictions. A novel machine learning framework for detecting COVID-19 symptomatic tweets in real-time in multiple languages is outlined. By collating the tweets from the previous 24 hours a time series of symptomatic tweets can be set up per geographic region. It is shown that, when compared with other traditional data sources, such as positive test results, ingesting tweet data can result in more consistent and accurate COVID-19 death predictions in the United States, United Kingdom and European and South American countries.

Acknowledgements

This thesis was supported by a EPSRC and ESRC studentship at the Centre for Doctoral Training on Quantification and Management of Risk and Uncertainty in Complex Systems Environments.

First of all I would like to thank my primary supervisor, Professor Simon Maskell and my secondary supervisor Dr. John Harris for mentoring me throughout the duration of my studies. Without their guidance and support this thesis would not have been possible.

I would also like to thank the whole signal processing research group at the University of Liverpool for making my time such an enjoyable experience. In particular Dr. Lee Devlin and Dr. Paul Horridge for their patience when responding to countless emails and explaining concepts related to signal processing in a manageable way. I would also like to give a special mention to Kelli Cassidy and Sara Parker for organising the research group. Thank you to the friends I made as part of the Risk and Uncertainty research group.

I would also like to thank Professor. Thomas Schön for hosting me for a week to discuss opportunities for future work which allowed me to experience Sweden and Dr. Jasmina Panovska-Griffiths for being a great mentor during my secondment at the UK Health Security Agency at such a difficult time during the COVID-19 pandemic.

I would also like to thank my family and friends. My parents for helping me financially, emotionally and getting me to think about things other than football. Catriona, Alex and Hannah for giving me much needed laughs during the past 5 years. I thank all my team mates at the Mersey Harps and friends in the Cream Team for distracting me from my work and reminding me of life outside of my PhD.

Finally I would like to thank Kira for her patience, support and guidance throughout my studies.

Contents

Abbreviations

The following abbreviations can be found throughout this thesis:

ABM	Agent Based Model
ABC	Approximate Bayesian Computation
ACF	Auto-Correlation Function
AD	Automatic Differentiation
BERT	Bidirectional Encoder Representations from Transformers
CRN	Common Random Numbers
CPRS	Continuous Ranked Probability Score
ESS	Effective Sample Size
FI	Fisher's Identity
HMC	Hamiltonian Monte Carlo
HPC	High Performance Computer
IACT	Integrated Auto-Correlated Time
ICU	Intensive Care Unit
JBC	Joint Bio-security Centre
LMC	Langevin Monte Carlo
LGSSM	Linear Gaussian State-Space Model
MCMC	Markov Chain Monte Carlo
MALA	Metropolis-Adjusted Langevin Algorithm

MHRW	Metropolis-Hastings Random Walk
NHS	National Health Service
NLP	Natural Language Processing
NEES	Normalised Estimation Error Squared
NUTS	No-U-Turn Sampler
NGE	Number of Gradient Evaluations
ODE	Ordinary Differential Equations
O/D	Origin/Destination
p-MCMC	Particle- Markov Chain Monte Carlo
p-HMC	Particle-HMC
p-NUTS	Particle-NUTS
PPL	Probabilistic Programming Language
ROW	Rest of the World
RMSE	Root Mean Square Errors
SBC	Simulation Based Calibration
SMC	Sequential Monte Carlo
SSM	State-space Model
SDE	Stochastic Differential Equations
SV	Stochastic Volatility
SVM	Support Vector Machine
SIR	Susceptible, Infected and Recovered
UK	United Kingdom
UKHSA	United Kingdom Health Security Agency
UoL	University of Liverpool
w.r.t	With Respect To
WIS	Weighted Interval Score
WHO	World Health Organisation

Probability Distributions

Distribution	**Notation**	**Parameters**	**Mass Function**
Binomial	$\mathcal{B}(n,p)$	$n \in \mathbb{N},\ p \in [0,1]$	$\binom{n}{k} p^k (1-p)^{n-k}$
Negative Binomial	$\mathcal{NB}(r,p)$	$r \in \mathbb{N}_+,\ p \in [0,1]$	$\binom{k+r-1}{k} p^r (1-p)^k$
Poisson	$\mathcal{P}(\lambda)$	$\lambda > 0$	$\frac{\lambda^k}{k!} e^{-\lambda}$

Table 1: Parameterisation of discrete probability distributions used in the analysis of this thesis.

Distribution	**Notation**	**Parameters**	**Density Function**
Gamma	$\mathcal{G}(\alpha,\beta)$	$\alpha > 0,\ \beta > 0$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$
Normal	$\mathcal{N}(\mu,\sigma)$	$\mu \in (-\infty,\infty),\ \sigma > 0$	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma^2}\right)}$
Uniform	$\mathcal{U}(a,b)$	$-\infty < \alpha < b < \infty$	$\begin{cases} \frac{1}{b-a}, & x \in [a,b] \\ 0, & \text{otherwise} \end{cases}$

Table 2: Parameterisation of continuous probability distributions used in the analysis of this thesis.

Chapter 1

Introduction

1.1 Epidemiological Modelling

Two methods for modelling diseases include: Agent-based models (ABMs) [1] which simulate interactions with individuals within the population based on a set of rules; and the compartmental approach of splitting the population into unobservable compartments, for example Susceptible, Infected and Recovered (SIR) [2] and allowing a fraction at every timestep t, to progress to the next compartment. Although compartmental models do simulate interactions with individuals in a population, they don't represent them explicitly like ABMs. The compartmental model was utilised as far back as 1665 in London to describe the rise of infected patients during the plague [3]. The model consists of three nonlinear ordinary differential equations (ODE) and a set of parameters which govern how quickly individuals progress through the compartments. The standard SIR model is described in Section 3.1.4 and contains two parameters, β and γ which are the infection and recovery rates, respectively. Knowing these parameters of a specific disease allows public health officials and researchers to ascertain the reproductive number R_t at time t. This metric quantifies the average number of secondary infections produced by a single infected person and can be calculated by

$$R_t = \beta/\gamma. \tag{1.1}$$

This gives an indication of how fast a disease is spreading through the population. A popular method for representing the traditional SIR model is in a state-space model

(SSM) framework. SSMs are a favoured method for representing the dependence of latent states in non-linear dynamical systems and have been widely used to model the dynamics of communicable diseases in populations of interest by fitting to time series data. A SSM consists of a state equation,

$$x_t \mid x_{t-1} \sim f(x_t \mid x_{t-1}, \theta), \tag{1.2}$$

which is parameterised by θ, where f is an arbitrary function that describes how the dynamical system moves from the previous state to the current state, x_t, at time t. The SSM also includes an observation equation

$$y_t \mid x_t \sim g(y_t \mid x_t, \theta), \tag{1.3}$$

where the arbitrary function g describes how the observation, y_t, is linked to the state equation. One of the advantages of the SIR model is its ability to model a wide range of communicable diseases such as Ebola [4], influenza [5], HIV/AIDS [6] and COVID-19 [7]. However, not all communicable diseases or infections manifest at the same rates. Extensions of the original SIR model attempt to account for this. For example the SEIR model contains an extra exposed compartment which describes individuals that have been exposed to the infection but are not themselves infectious yet. This mean latent period for the disease is accounted for by an additional parameter, δ. Some examples can be found here [8, 9, 10].

Particle filters [11, 12, 13] are a popular method for inferring the time-dependent hidden states of SSM and have been previously applied to compartmental disease transmission models [14, 15, 16, 17]. Particle filters estimate the states of an SSMs via a set of weighted samples (particles). Each sample is a different hypothesis of the state of the system and the combination of these samples at every timestep is an estimation of the posterior distribution over the states. One of the benefits to using a particle filter is the ability to use stochastic models such as stochastic differential equations (SDE) in place of an ODE. For example, noise parameters can be added to the SIR ODE model as seen in [14, 15, 16, 17]. Another popular method for stochastically modelling epidemics is described by Reed-Frost [18] and assumes that people becoming infected with a disease follows a Binomial distribution [19, 20].

1.2 Calibration

Calibration of epidemiological models using disease specific data is important when striving to understand how a disease evolves with time. Inferring parameters such as the effective contact and recovery rates, especially at the beginning of an outbreak, is vital for public health officials when assessing how a disease spreads through a population. Two overarching methods used for estimating parameters in the context of epidemiological models, as outlined in a recent systematic review [21], are optimisation and sampling algorithms. Optimisation algorithms include grid search and iterative, descent-guided optimisation while sampling methods include MCMC techniques (see Section 3.3), p-MCMC (see Section 3.4), Approximate Bayesian Computation (ABC) and history matching. One difference between these two groups of techniques is that optimisation algorithms typically find point estimates of the parameters[1] while sampling methods can produce full Bayesian parameter estimates which in-turn capture parameter uncertainty. A comparison of software packages that include some of these optimisation and sampling algorithms can be found here [22].

MCMC methods sample from a probability distribution where the current sample depends on the previous. Sampling in this manner produces a Markov Chain which gives a distribution around the quantity being estimated. Bayesian calibration of the SSMs in (1.2) and (1.3) is undertaken with the goal of estimating the parameter posterior distribution $p(\theta|y)$. This involves finding the set θ that best represents the data, y, using Bayes theorem:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} = \frac{p(y|\theta)p(\theta)}{\int_\theta p(y|\theta)p(\theta)d\theta}, \tag{1.4}$$

where $p(\theta)$ is the prior, $p(y|\theta)$ the likelihood and $p(y)$ the evidence. The likelihood and the prior are easily calculated if their explicit forms are known. However, the integral in (1.4) often becomes intractable in high dimensions. Therefore, the posterior is typically estimated up to a normalisation constant, given by the integral.

The time until stationarity of the Markov chain is reached is dependent on its initial starting point. It is therefore common practice to discard the first portion of the Markov chain as *burn-in*. *Burn-in* is sometimes referred to as *warm up* which allows the sampler to get to an acceptable part of the parameter space to start sampling. Hyper-parameter

[1] We note that optimisation algorithms can indeed provide uncertainty through methods such as likelihood ratio.

tuning also occurs which ensures sampling is efficient (eg the mass matrix and step size used in NUTS). Ensuring convergence to the correct posterior distribution, the Markov chains are ergodic. Three popular MCMC algorithms are described in Section 3.3.

In the well understood MHRW proposal a set of parameters θ' is drawn from a proposal distribution which is commonly chosen to be a Gaussian centered around the current position, θ, and a variance parameter, also known as step size, which is chosen by the user. An accept or reject step is introduced that determines whether to accept θ' as part of the Markov chain or reject and revert back to θ. Such proposals can struggle to enable the Markov chain to reach the stationary distribution when estimating large numbers of parameters. A related issue can occur with Gibbs samplers when the correlation between parameters is high. These issues can result in the sampler getting stuck in local maxima within the target distribution, $\pi(\theta)$. HMC is a a gradient based sampler and is better at proposing samples than a random-walk proposal due to the use of gradients. It was first developed in the late 1980s [23] and in the last decade has become a popular approach when implementing MCMC [24, 25]. It is an approach that simulates from a problem-specific Hamiltonian system to generate samples. HMC is effective when the target distribution is complex or multi-modal but is sensitive to hyperparameters which have to be determined by the user. The No-U-Turn Sampler (NUTS) [26] is an adaptive version of HMC which automates the selection of these hyperparameters. Probabilistic programming languages (ppls) such as Stan [27] and PyMC3 [28] are tools that have been developed to allow users to define and make inferences about probabilistic models using NUTS.

P-MCMC [29] combines the particle filter and MCMC to make state and parameter estimates by performing Bayesian inference on non-linear non-Gaussian scenarios where standard MCMC methods can fail. P-MCMC has been used to infer parameters of simulated genealogies and time series data [30], a dengue outbreak [31], non-communicable diseases in Egypt [32], cholera in Bangladesh [33], COVID-19 in the United Kingdom (UK) [34], the 2009 A/H1N1 pandemic in England [35] and an ABM in [36]. It has been suggested in [37, 38] that a potential reason why p-MCMC has yet to become a mainstream method for modelling epidemics is due to the complexity of the mathematics behind the algorithm: [38] make the methodology more accessible by providing a step-by-step tutorial. The standard proposal used in calibration of epidemiological models when using p-MCMC is the MHRW [30, 32, 33, 34, 35, 36, 38] which inherits the same drawbacks as explained above for MCMC.

In order to use gradient based proposals within p-MCMC, the gradient of the log-

likelihood w.r.t θ needs to be calculated. In order to obtain these gradients the particle filter needs to differentiated. However, it has been noted in [39, 40, 41] that the stochastic nature of both the sampling and resampling steps, that are inherently part of the particle filter, are not differentiable. The *reparameterisation trick* was proposed in [42] to reformulate the sampling operation into a differentiable function by sampling a noise vector in advance and defining the likelihood for θ as being a deterministic function of this sampled noise vector. However, resampling remains problematic. Differentiable resampling that is compatible with Automatic Differentiation (AD) has been an active area of research in machine learning. These include: leveraging the ideas of optimal transport to produce a continuous approximation to the discrete distribution [43], a differentiable approximation based on importance sampling (referred to as *soft-resampling* [44]) and continuous relaxations of discrete distributions [45, 46] (referred to as the Gumbel-Softmax). An overview and comparison of these methods can be found here [47].

Extensions of the original p-MCMC algorithm described in Section 3.4 have focused on including gradient information when proposing new parameters. Reference [48] shows how to estimate the score (gradient) of the log-likelihood and the observed information matrices at θ in SSMs using particle filter methods. The two methods proposed run with computational complexity $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$, respectively. The first has a linear computation cost but the performance decreases over time. The second has a computation cost that increases quadratically with the number of particles N but performance does not deteriorate over time, with [49] theoretically substantiating this claim. Reference [50] built on this work to compute these terms with computational complexity $\mathcal{O}(N)$ and avoids the quadratically increasing variance caused by particle degeneracy. In [51, 52, 53, 54] the authors utilise the previous work of [50] to recursively estimate the score (gradient) of the log-likelihood at θ. References [51] and [52] include Langevin Monte Carlo (LMC) methods seen in [55] whilst [53] and [54] include first- and second-order Hessian information about the posterior in the proposal. Use of the Hessian is shown to improve the mixing of the Markov chain at the stationary phase and decrease the length of *burn-in*. However, calculating a $d \times d$ matrix of the second-order partial derivatives can become infeasible when the dimensionality, d, becomes large. While [53], [54] do mention using HMC dynamics within p-MCMC, no implementation of this approach has been described in the literature.

1.3 Evaluating Forecasts

Short-term forecasts of certain metrics can be useful when determining how well a model represents reality. Calibration techniques can be compared in terms of accuracy e.g. death forecasts can be compared against true deaths. By January 2020, new cases of the novel coronavirus (COVID-19) had been seen throughout Asia, and by the time the World Health Organisation (WHO) declared a global pandemic in March 2020, COVID-19 had spread to over 100 countries. Therefore, it was imperative to establish reliable data feeds relating to the pandemic so researchers and analysts could model the ongoing spread of the disease and inform decision-making by government and public health officials. These datasets and models must be open-source to facilitate collaboration between researchers and allow for published results to be replicated and scrutinised. A popular interactive dashboard that collates total daily counts of confirmed cases and deaths for countries, and in some cases, regions within countries exists here [56]. These variables are traditionally used to calculate metrics such as the reproduction number R_t, which is vital in understanding both the number of people on average an infected person infects and the infection growth rate or daily rate of new infections. The quality of the metrics calculated is heavily dependent on the model and ingested data. Challenges do exist when ingesting multiple data streams at once [57].

While it is possible to use deaths to provide a reliable information feed, the latency of data derived from deaths is significant. In the context of COVID-19, confirmed cases derived from positive test results potentially provide a lower latency data feed. However, the sampling of those tested varies with time and the reason for testing is often not recorded. Hospital admissions typically occur around 1-2 weeks after infection and can be considered out of date in relation to the time of initial infection [58]. The extent to which these issues are problematic is likely to vary over time and between countries. For example at the beginning of the COVID-19 pandemic it took months for a reliable test which was available to the public to be created so information about the spread was limited. Therefore, certain countries may have had different testing capacities compared with others. Tweets relating to influenza have been seen to correlate with actual influenza counts [59, 60, 61]. A pipeline for collecting and analysing COVID-19 tweets could be set up in a short amount of time and can be scaled up to multiple countries.

Chapter 2

Motivation, Contribution and Thesis Structure

In the subsequent paragraphs, the main motivations and contributions of this thesis are outlined and a description of the published works provided.

2.1 Chapter 3

Chapter 3 provides a summary of the main topics utilised throughout this thesis.

2.2 Chapter 4

The main motivation of this thesis was to provide an extension of p-MCMC that uses the gradient of the log-likelihood w.r.t θ within the MCMC proposal. A recent published journal article proposed using first- and second-order Hessian information about the posterior in a MALA proposal [54]. This prompted a meeting at Uppsala University, Sweden with Professor. Thomas Schön, a co-author of [54]. We discussed how to develop their methods to include HMC and NUTS proposals. This led to a novel implementation of the differentiable particle filter which is described in Chapter 4. I used initial derivations provided by Dr. Paul Horridge and Professor. Simon Maskell and calculated the gradients from a particle filter which modelled different SSMs. These gradients enable use of gradient based proposals such as MALA, HMC and NUTS. The work on which Chapter 4 is based is published in the journal IEEE Transactions of Signal Processing [47]:

- **C. Rosato**., L. Devlin., V. Beraud., P. Horridge., T. B. Schön. and S. Maskell. "Efficient Learning of the Parameters of Non-Linear Models Using Differentiable Resampling in Particle Filters," in IEEE Transactions on Signal Processing, vol. 70, pp. 3676-3692, 2022, doi: 10.1109/TSP.2022.3187868.

An application of the differentiable particle filter with NUTS, applied to epidemiological models, is presented in Section 4.7.3. A comparison is made between the MHRW proposal, the commonly chosen method when tracking diseases, and NUTS, when estimating the parameters of the SIR and SEIR models. I coded the entirety of these algorithms, collected and analysed the experimental results. The published version of these results appears in [62]:

- **C. Rosato**., J. Harris., J. Panovska-Griffiths. and S. Maskell, "Inference of Stochastic Disease Transmission Models Using Particle-MCMC and a Gradient Based Proposal," 2022 25th International Conference on Information Fusion (FUSION), 2022, pp. 1-8, doi: 10.23919/FUSION49751.2022.9841249.

2.3 Chapter 5

The version of NUTS used in Chapter 4 is the original algorithm described in [26]. The NUTS algorithm has been extensively developed in Stan and PyMC3, resulting in more efficient samplers. Chapter 5 outlines a number of these developments, in addition to a framework for calculating the gradients of a particle filter using AD and PyTorch. These gradients can be used within a PyMC3 implementation of NUTS. A comparison between NUTS and MHRW, when estimating the parameters of three SSMs using real datasets, is presented. This work will be extended and submitted to a signal processing and machine learning conference.

2.4 Chapter 6

Chapter 6 outlines a novel extended SEIR epidemiological model used to infer the spread of COVID-19 in England. This model will be referred to as the University of Liverpool (UoL) model throughout the remainder of this thesis. This work is published in [63]:

- R. E. Moore., **C. Rosato**. and S. Maskell. "Refining epidemiological forecasts with simple scoring rules." Philosophical Transactions of the Royal Society A 380.2233 (2022): 20210305.

My contribution to this work outlines how simple scoring rules can detect shortcomings in a model when estimating COVID-19 deaths. In particular, using the scoring rule Normalised Estimation Error Squared (NEES) it is possible to observe whether predictions are over-confident or over-cautious. The model was used by the United Kingdom Health Security Agency (UKHSA) (formerly known as the Joint Bio-security Centre (JBC)) to provide weekly estimates of the reproductive number [64]. A brief summary of the UoL model is published on the UK government website [65].

2.5 Chapter 7

During the initial phase of the epidemic, the availability of accurate and up-to-date data sources was essential. One such method of data collection, previously used to detect disease outbreaks such as measles and influenza, is via social media. In particular, Twitter and the content of individual tweets has been found to be effective when detecting and localising outbreaks of diseases in real-time. Chapter 7 outlines a method to detect symptomatic COVID-19 tweets by training machine learning algorithms. Using the geo-location attached to each tweet, time series consisting of aggregated counts can be collated and attributed to a given country or region within a country. In the analysis of this chapter, 50 U.S states, 13 Latin American countries, 2 European countries and the 7 National Health Service (NHS) areas in the UK were considered. This analysis necessitated machine learning classifiers trained in English, Spanish, Italian, German and Portuguese. The flexibility of the UoL model allows for different data sources to be ingested when making inferences pertinent to the spread of COVID-19. Chapter 7 focuses on the usefulness of different publicly available data sources when making predictions of COVID-19 related deaths. The work on which Chapter 7 is based is published in the journal Information [66]:

- **C. Rosato**., R. E. Moore., M. Carter., J. Heap., J. Harris., J. Storopoli. and S. Maskell. "Extracting Self-Reported COVID-19 Symptom Tweets and Twitter Movement Mobility Origin/Destination Matrices to Inform Disease Models.", Information 2023, 14, 170., doi: 10.3390/info14030170

Chapter 3

Technical Information

The aim of this chapter is to introduce the main topics utilised throughout this thesis. These topics include SSMs and epidemiological models, particle filters and parameter estimation techniques such as MCMC and p-MCMC.

3.1 State-Space Models

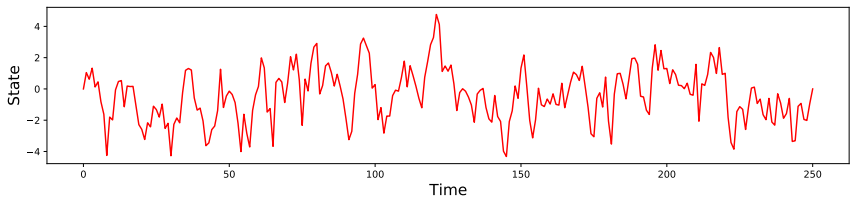
SSMs have been used to model dynamical systems in a wide range of research fields (see [67] for numerous examples). They are represented by two stochastic processes, $\{x_t\}_{t\geq 0}$ and $\{y_t\}_{t\geq 0}$. The hidden state, x_t, evolves according to a Markov process which is determined by $p\left(x_t|x_{t-1},\theta\right)$, and the observation is given by y_t. The model can then be defined such that

$$x_t|x_{t-1} \sim \ p\left(x_t|x_{t-1},\theta\right) \tag{3.1}$$

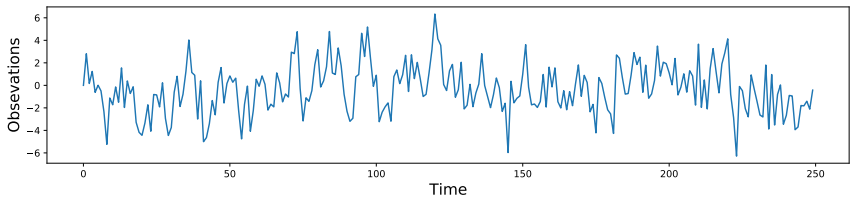
and

$$y_t|x_t \sim \ p\left(y_t|x_t,\theta\right), \tag{3.2}$$

where the arbitrary functions f and g in (1.2) and (1.3) are given by probability distributions. The density of the initial latent state x_0 is denoted $\mu_\theta(x_0)$. The SSM is parameterised by an unknown vector of parameters θ contained in the parameter space Θ. The transition and observation densities are given by (3.1) and (3.2), respectively. The SSMs used as examples in the chapters of this thesis are presented below.



(a)



(b)

Figure 3.1: (a) The latent state and (b) observations for the LGSSM in Section 3.1.1 for T =250 and true parameter values $\theta = \{0.75, 1.2, 1\}$.

3.1.1 Linear Gaussian State-Space Model

The linear Gaussian state-space model (LGSSM) is one of the simplest SSMs. The latent state and observation processes evolve according to a Gaussian distribution. The following example of an LGSSM is presented in [68]:

$$x_t \mid x_{t-1} \sim \mathcal{N}\left(x_t; \phi x_{t-1}, \sigma_v^2\right), \tag{3.3}$$

$$y_t \mid x_t \sim \mathcal{N}\left(y_t; x_t, \sigma_e^2\right), \tag{3.4}$$

where $\theta = \{\phi, \sigma_v, \sigma_e\}$ are parameters with prior densities Normal$(0, 1)$, Gamma$(1, 1)$ and Gamma$(1, 1)$, respectively. The subplots (a) and (b) in Figure 3.1 show the latent state and observations, respectively.

3.1.2 Stochastic Volatility Model

Stochastic volatility (SV) models are widely used to evaluate financial prices and securities [69], since the variance of the latent process changes over time and is not constant. The state x_t is defined to be a monotonic function of the observation noise. SV models have been represented by SSMs extensively in the literature. One example, presented in [68], is

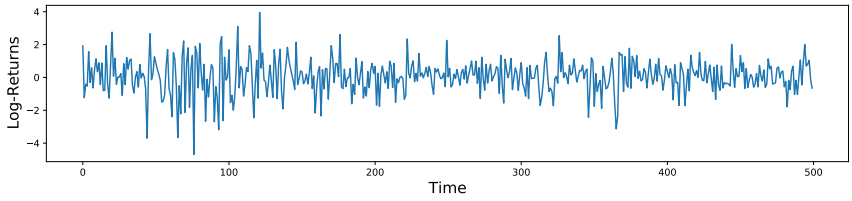


Figure 3.2: Daily log-returns from the stochastic volatility model.

defined as follows:

$$x_0 \sim \mathcal{N}\left(x_0; \mu, \frac{\sigma_v^2}{1-\phi^2}\right), \tag{3.5}$$

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \mu + \phi\left(x_t - \mu\right), \sigma_v^2\right), \tag{3.6}$$

$$y_t \mid x_t \sim \mathcal{N}\left(y_t; 0, \exp\left(x_t\right)\right), \tag{3.7}$$

where $\theta = \{\mu, \phi, \sigma_v\}$ are parameters with prior densities Normal$(0, 1)$, Normal$(0, 1)$ and Gamma$(2, 10)$, respectively. The observations (log-returns), y_t are then given by

$$y_t = 100 \log\left[\frac{s_t}{s_{t-1}}\right] = 100\left[\log\left(s_t\right) - \log\left(s_{t-1}\right)\right], \tag{3.8}$$

where s_t is the closing price. The daily log-returns corresponding to the closing prices of the NASDAQ OMXS30 index are presented in Figure 3.2, where the index gives a weighted average of the 30 most traded stocks at the Stockholm stock exchange. The data is extracted from Quandl[1] for the period between January 2, 2012 and January 2, 2014.

3.1.3 Earthquake Model

This example considers the annual number of earthquakes that occurred between 1900 and 2007 that reached over seven on the Richter scale.[2] A time series representation of the data is presented in Figure 3.3. The SSM is as described in [54, 70] and is presented below:

$$x_{t+1} \mid x_t \sim \mathcal{N}\left(x_{t+1}; \phi x_t, \sigma\right), \tag{3.9}$$

$$y_t \mid x_t \sim \mathcal{P}\left(y_t; \mu \exp\left(x_t\right)\right), \tag{3.10}$$

[1]The data can be downloaded from `https://data.nasdaq.com/data/NASDAQOMX/OMXS30`
[2]https://www.usgs.gov/programs/earthquake-hazards/earthquakes

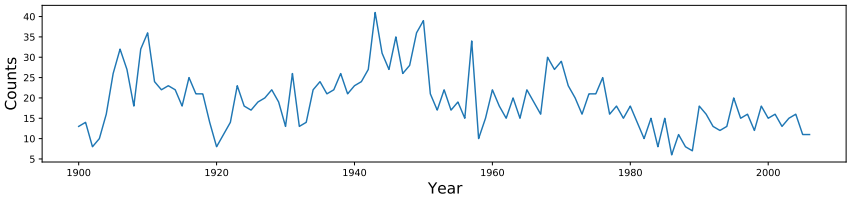


Figure 3.3: Earthquake counts.

where the distribution $\mathcal{P}$ corresponds to the Poisson distribution. The parameters $\theta = [\phi, \sigma, \mu]$, where $|\phi| < 1, \sigma > 0$ and $\mu > 0$, each have uniform priors. Note that these priors are taken from [54] and are improper.

3.1.4 SIR Epidemiological Model

3.1.4.1 Continuous Model

The SIR epidemiological model simulates the spread of disease through a population of size P. The population is split into three unobservable compartments. At each timestep, t, proportions of the population move from the susceptible to infected and infected to recovered compartments. The number of individuals in each compartment are given by the nonnegative variables S_t, I_t and R_t, such that $S_t + I_t + R_t = P$. The model assumes that no births or deaths occur throughout the simulation period, i.e. that the population P is constant, homogeneous mixing of infected and susceptible compartments and homogeneous age and sex across the population. Chapter 6 outlines how the SIR model can be extended to account for more realistic scenarios.

The system of differential equations simulating disease propagation under the SIR model is as follows:

$$\frac{dS}{dt} = -\frac{\beta IS}{P}, \tag{3.11}$$

$$\frac{dI}{dt} = \frac{\beta IS}{P} - \gamma I, \tag{3.12}$$

$$\frac{dR}{dt} = \gamma I, \tag{3.13}$$

with discrete time approximation given by

$$S_{t+1} = S_t - \frac{\beta I_t S_t}{P}\Delta t, \tag{3.14}$$

$$I_{t+1} = I_t + \frac{\beta I_t S_t}{P} - \gamma I_t \Delta t, \tag{3.15}$$

$$R_{t+1} = P - S_{t+1} - I_{t+1}, \tag{3.16}$$

where Δt is the interval between timesteps. The effective transmission rate of the disease and the mean recovery rate are given by β and γ, respectively. These parameters govern how quickly the population moves through the compartments.

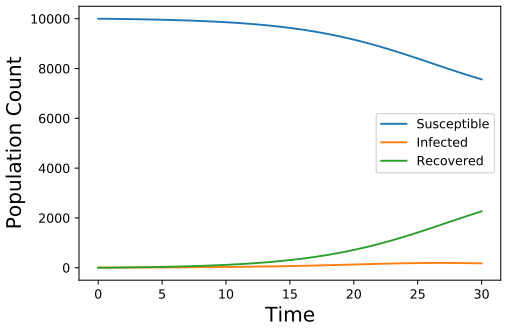
The reproductive number is calculated via (1.1). A value less than 1 indicates that the virus is dying out and a value greater than 1 that the disease is continuing to spread throughout the population. This is exemplified in the subplots of Figure 3.4. As R_t increases, more of the susceptible population become infected and subsequently recover within a fixed observation period. The peak of the epidemic is consequently observed at an earlier point in time. During an outbreak of a disease, one of the main goals for public health officials is to flatten the infectious curve. A number of methods for achieving this in the context of COVID-19, are presented in [71].

It is possible to include stochasticity within the SIR model by adding stochastic fluctuations to the disease dynamics. Inclusion of a noise term, ϵ_x, for each time-varying parameter, x, mimics the randomness of the interactions between individuals in a population [16, 17]. The noise terms corresponding to each parameter are independent and are drawn from $\epsilon_x \sim \mathcal{N}(0, \sqrt{x}/P)$, such that, for example, $\epsilon_\beta \sim \mathcal{N}(0, \sqrt{\beta}/P)$. The discrete time approximation of the stochastic SIR model is then given by

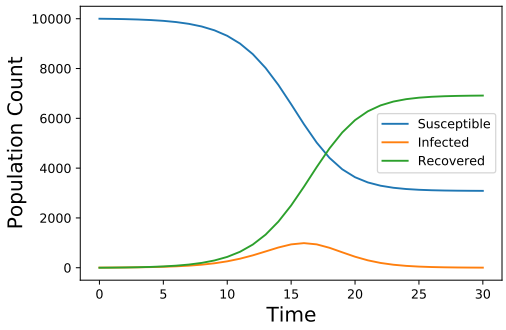
$$S_{t+1} = S_t - \frac{\beta I_t S_t}{P} + \epsilon_\beta \Delta t, \tag{3.17}$$

$$I_{t+1} = I_t + \frac{\beta I_t S_t}{P} - \gamma I_t \epsilon_\beta + \epsilon_\gamma \Delta t, \tag{3.18}$$

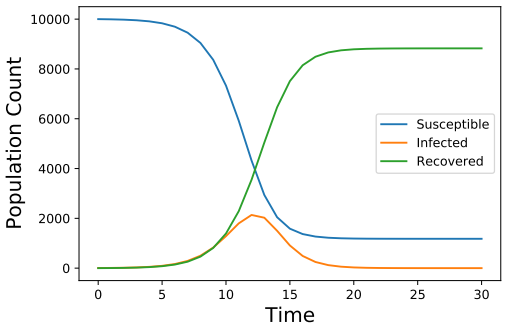
$$R_{t+1} = P - S_{t+1} - I_{t+1}. \tag{3.19}$$



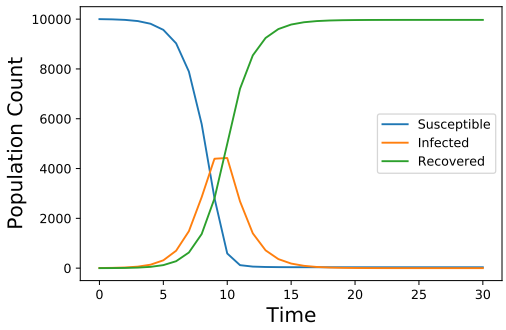
(a) $\beta = 1.1$, $\gamma = 0.9$, $R_t = 1.2$



(b) $\beta = 1.3$, $\gamma = 0.8$, $R_t = 1.6$



(c) $\beta = 1.5$, $\gamma = 0.7$, $R_t = 2.1$



(d) $\beta = 1.8$, $\gamma = 0.5$, $R_t = 3.6$

Figure 3.4: Epidemic curves for varying β and γ for $P = 10000$.

3.1.4.2 Discrete Model

A further method for incorporating stochasticity in the SIR model is to specify the number of individuals leaving a compartment as a binomially distributed random variable. The binomial probability, p, is defined as the rate at which individuals leave a compartment, given by

$$p(S \to I) = 1 - e^{\frac{-\beta i_t s_t}{P}}, \tag{3.20}$$

$$p(I \to R) = 1 - e^{-\gamma}, \tag{3.21}$$

where $p(S \to I)$ and $p(I \to R)$ denote the probability of leaving the susceptible and infected compartments, respectively. The corresponding binomial distributions are

$$n(S \to I) \sim \text{Binomial}(S_{t-1}, p(S \to I)), \tag{3.22}$$

$$n(I \to R) \sim \text{Binomial}(I_{t-1}, p(I \to R)), \tag{3.23}$$

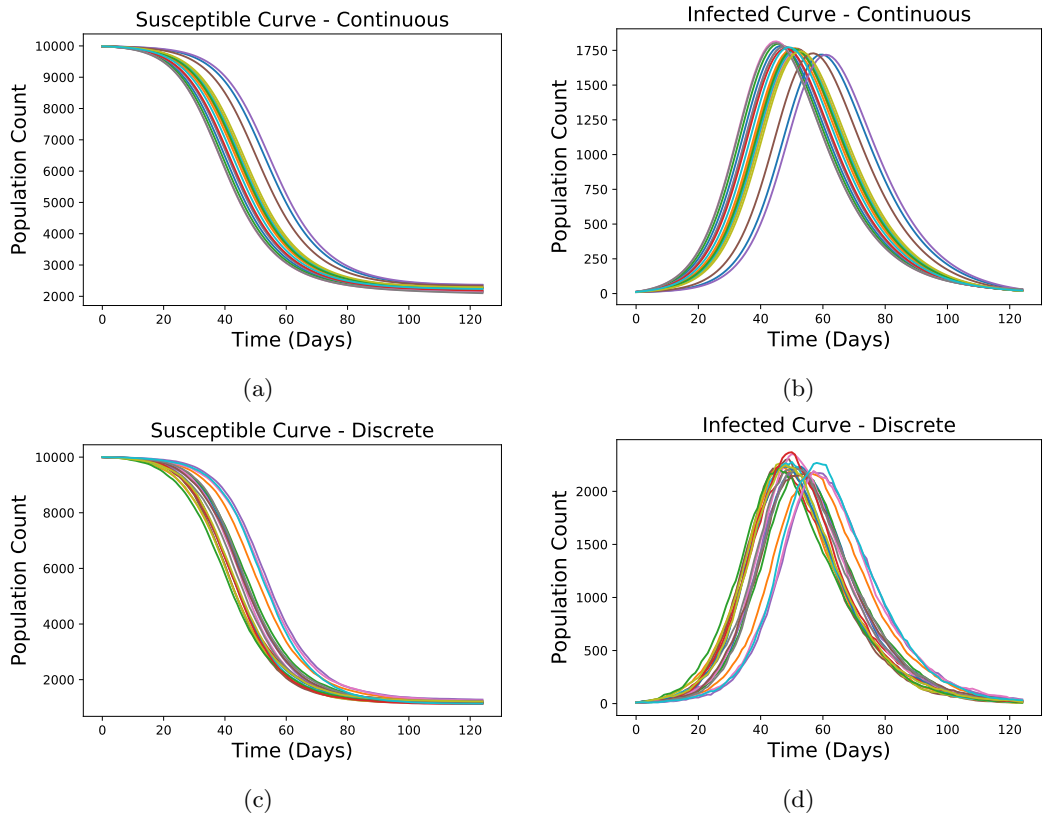
where $n(S \to I)$ and $n(I \to R)$ denote the total number of individuals leaving the susceptible and infected compartments, respectively. The complete discrete, stochastic SIR model is therefore presented as

$$S_{t+1} = S_t - n(S \to I), \tag{3.24}$$

$$I_{t+1} = I_t + n(S \to I) - n(I \to R), \tag{3.25}$$

$$R_{t+1} = P - S_{t+1} - I_{t+1}. \tag{3.26}$$

Figure 3.5 shows how the incorporation of stochasticity generates different realisations of epidemic curves within the population.



(a) (b) (c) (d)

Figure 3.5: 20 susceptible and infected curves for the continuous (top) and discrete (bottom) models outlined in Sections 3.1.4.1 and 3.1.4.2, respectively, for $P = 10000$, $I_0 = 10$, $\beta = 0.254$ and $\gamma = 0.111$.

3.1.4.3 Observation Equation

As the compartments of the SIR model are unobservable, in order to estimate the true level of infection within the population, an observation equation (often referred to as the likelihood) can be used to link the syndromic surveillance data to the infected compartment. Relevant examples of syndromic data include prescription sales, disease specific positive test results, general practice and hospital admissions data and social media posts mentioning disease specific symptoms.

Specification of the likelihood is dependent on the surveillance data. One example is to define the log of the observations, y_t, such that

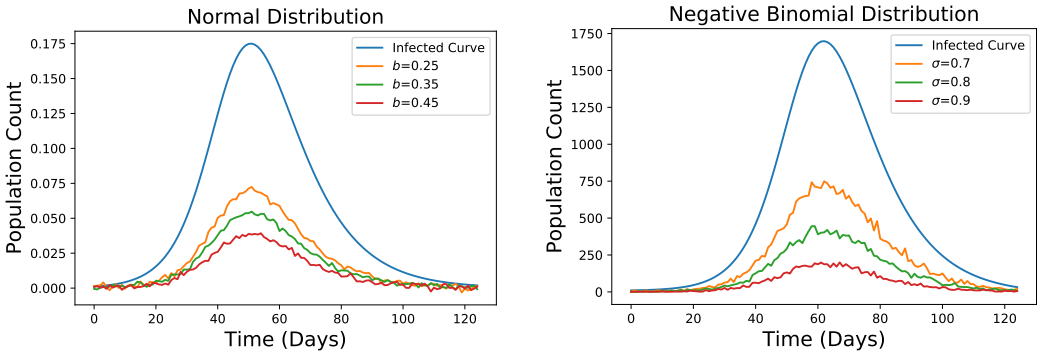
$$\log y_t \sim \mathcal{N}\left(i_t, \sigma^2\right), \tag{3.27}$$

where i_t is the proportion of infected individuals at time t [16, 17]. Another commonly used approach is to model the observations using the negative binomial distribution:

$$y_t \sim \mathcal{NB}\left(i_t, \sigma\right), \tag{3.28}$$

where σ is the over dispersion parameter [72] [3]. The size of the peak in the syndromic data is governed by the parameters within the observation equations (3.27) and (3.28). Figure 3.6 exemplifies this in subplots (a) and (b), where sensitivity analysis on the parameters b and σ is presented. When considering real disease outbreaks it is essential to calibrate epidemiological models to surveillance datasets. Information on undertaking this using MCMC and p-MCMC is presented in Sections 3.3 and 3.4, respectively.

[3] This is the alternative parameterisation of the negative binomial distribution as defined by the Stan Development Team [73].



(a) (b)

Figure 3.6: Realisations of the infected curve and the corresponding (a) log of observations drawn from (3.27) for $b = 0.25, 0.35, 0.45$, (b) observations drawn from (3.28) for $\sigma = 0.7, 0.8, 0.9$.

3.2 Particle Filter

Particle filters have been used in many areas of research, such as finance [74], disease modelling [75] and multiple target tracking [76], to infer time-dependent hidden states. One benefit of using a particle filter is its ability to model SSMs that are nonlinear and non-Gaussian. A method for including stochastic terms within the SIR model through implementation of a particle filter is now presented.

To formulate the problem, consider t timesteps, where data is obtained at each increment of time. Let the transition and observation densities be parameterised by θ (which has n_θ dimensions), such that

$$p\left(y_{1:t}, x_{1:t}|\theta\right) = p(y_1|x_1, \theta)p\left(x_1|\theta\right) \times \prod_{\tau=2}^{t} p\left(y_\tau|x_\tau, \theta\right) p\left(x_\tau|x_{\tau-1}, \theta\right), \tag{3.29}$$

where the state x_t has n_x dimensions and the set of all data obtained up to time t, $y_{1:t}$, and the state sequence, $x_{1:t}$, grow over time. If θ is known, a (conventional) particle filter can be used as follows.

At every timestep t, the particle filter draws N samples (particles) from a proposal distribution, $q\left(x_{1:t}|y_{1:t}, \theta\right)$, which is parameterised by the sequence of states and observations. The samples are statistically independent and each represents a different hypothesis of the sequence of states of the system. The ith sample has an associated weight, $w_t^{(\theta,i)}$,

which indicates the relative importance of each of the corresponding samples. A set of N particles can then be represented as $\left\{x_{1:t}^{(\theta,i)}, w_t^{(\theta,i)}\right\}_{i=1}^{N}$. At $t = 0$, the weights are fixed at $1/N$.

The proposal distribution is constructed recursively by

$$q\left(x_{1:t}|y_{1:t},\theta\right) = q\left(x_1|y_1,\theta\right)\prod_{\tau=2}^{t} q\left(x_\tau|x_{\tau-1},y_\tau,\theta\right), \tag{3.30}$$

such that an estimate with respect to the joint distribution, $p\left(y_{1:t},x_{1:t}|\theta\right)$, is posed as follows:

$$\int p\left(y_{1:t},x_{1:t}|\theta\right) f\left(x_{1:t}\right) dx_{1:t} \approx \frac{1}{N}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)} f\left(x_{1:t}^{(i)}\right). \tag{3.31}$$

This is an unbiased estimate, where

$$w_{1:t}^{(\theta,i)} = \frac{p\left(y_1|x_1^{(\theta,i)},\theta\right)p\left(x_1^{(\theta,i)}|\theta\right)}{q\left(x_1^{(\theta,i)}|y_1,\theta\right)} \times \frac{\prod_{\tau=2}^{t} p\left(y_\tau|x_\tau^{(\theta,i)},\theta\right)p\left(x_\tau^{(\theta,i)}|x_{\tau-1}^{(\theta,i)},\theta\right)}{\prod_{\tau=2}^{t} q\left(x_\tau^{(\theta,i)}|x_{\tau-1}^{(\theta,i)},y_\tau,\theta\right)} \tag{3.32}$$

$$= w_{1:t-1}^{(\theta,i)} \frac{p\left(y_t|x_t^{(\theta,i)},\theta\right)p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)}{q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right)} \tag{3.33}$$

is a recursive formulation for the unnormalised weight, $w_{1:t}^{(\theta,i)}$. The incremental weight is determined by

$$\sigma\left(x_t^{(\theta,i)},x_{t-1}^{(\theta,i)},\theta\right) = \frac{p\left(y_t|x_t^{(\theta,i)},\theta\right)p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)}{q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},y_t\right)}, \tag{3.34}$$

where for $t = 1$,

$$\sigma\left(x_{1:1}^{(\theta,i)}\right) = \frac{p\left(y_1|x_1^{(\theta,i)},\theta\right)p\left(x_1^{(\theta,i)}|\theta\right)}{q\left(x_1^{(\theta,i)}|y_1\right)}. \tag{3.35}$$

3.2.1 Choice of Proposal

Three common approaches for specifying the proposal distribution are:

1. Using the transmission model as the proposal:

$$q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, y_t\right) = p\left(x_t^{(i)}|x_{t-1}^{(\theta,i)}, \theta\right), \tag{3.36}$$

which simplifies the weight update to

$$w_{1:t}^{(\theta,i)} = p\left(y_t|x_t^{(\theta,i)}, \theta\right) w_{1:t-1}^{(\theta,i)}. \tag{3.37}$$

2. Using the "optimal" proposal:

$$q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, y_t\right) = p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, y_t\right) \tag{3.38}$$

with weights updated according to

$$w_{1:t}^{(\theta,i)} = p\left(y_t|x_{t-1}^{(\theta,i)}, \theta\right) w_{1:t-1}^{(\theta,i)}. \tag{3.39}$$

This approach is only possible in certain situations. Note that the term "optimal" in the context of the particle proposal means that the variance of the incremental particle weights at the current timestep is minimized. This variance is, in fact, zero since the weight in (3.39) is independent of x_t (as explained in [77]).

3. Using the Unscented Transform to approximate the optimal proposal, as explained in [78].

3.2.2 Estimation with Respect to the Posterior

It is often the case that estimates with respect to the posterior, $p\left(x_{1:t}|y_{1:t}, \theta\right)$, are needed. This can be done using the fact that

$$\int p\left(x_{1:t}|y_{1:t}, \theta\right) f\left(x_{1:t}\right) dx_{1:t} = \int \frac{p\left(y_{1:t}, x_{1:t}|\theta\right)}{p\left(y_{1:t}|\theta\right)} f\left(x_{1:t}\right) dx_{1:t}, \tag{3.40}$$

where

$$p\left(y_{1:t}|\theta\right) = \int p\left(y_{1:t}, x_{1:t}|\theta\right) dx_{1:t} \approx \frac{1}{N} \sum_{i=1}^{N} w_{1:t}^{(\theta,i)} \tag{3.41}$$

in line with (3.31). Then,

$$\int p\left(x_{1:t}|y_{1:t},\theta\right) f\left(x_{1:t}\right) dx_{1:t} \approx \frac{1}{\frac{1}{N}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)}} \frac{1}{N} \sum_{i=1}^{N} w_{1:t}^{(\theta,i)} f\left(x_{1:t}^{(\theta,i)}\right) \tag{3.42}$$

$$= \sum_{i=1}^{N} \tilde{w}_{1:t}^{(\theta,i)} f\left(x_{1:t}^{(\theta,i)}\right), \tag{3.43}$$

where

$$\tilde{w}_{1:t}^{(\theta,i)} = \frac{w_{1:t}^{(\theta,i)}}{\sum_{j=1}^{N} w_{1:t}^{(\theta,j)}} \tag{3.44}$$

are the normalised weights.

As (3.43) is a ratio of estimates, it is a biased estimate, in contrast to (3.31).

3.2.3 Resampling

The algorithm described in the preceding sections is the sequential importance sampling (SIS) algorithm. As time evolves, the normalised weights are increasingly skewed such that one of the weights (3.44) becomes close to unity, while the others approach zero. This is inevitable and cannot be avoided [79].

It is often suggested that monitoring the number of effective samples, N_{eff}, where

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{w}_{1:t}^{(\theta,i)}\right)^2}, \tag{3.45}$$

can be used to identify the need to resample. There are many resampling methods, some of which are outlined and evaluated in [80]. However, they all share the same purpose—to stochastically replicate particles with higher weights whilst eliminating ones with lower weights. Multinomial resampling is commonly used in practice. This involves drawing from the current particle set N times, proportionally to the associated weights of the particles. The associated distribution is defined by

$$\tilde{w}_{1:t}^{(\theta,i)} \quad \text{for} \quad i = 1, \ldots, N. \tag{3.46}$$

To keep the total unnormalised weight constant (such that the approximation (3.41) is the same immediately before and after resampling), each newly-resampled sample is

assigned an unnormalised weight

$$\frac{1}{N}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)}. \tag{3.47}$$

This is such that the normalised weights after resampling are $\frac{1}{N}$.

Algorithm 1: Particle Filter

Input: θ, $y_{1:T}$

Initialise: x_0^i, $\log(w_0^i)$

for $t = 1, \ldots, T$ **do**

  Calculate the number of effective samples, N_{eff}, using (3.45).

  If $N_{eff} < N/2$, resample x_{t-1}^i as described in Section 3.2.3.

  Sample the new particles x_t^i using (3.2.1)

  Evaluate the new log weights $\log w_{1:k}^i$ using (3.34).

end

Evaluate the final log-likelihood, $\log p(y_{1:T}|\theta)$ using (3.41)

3.3 Markov Chain Monte Carlo

MCMC methods allow for approximate inference of the posterior in (1.4) that does not require knowledge of the normalisation constant by considering

$$p(\theta|y) \propto p(y|\theta)p(\theta). \tag{3.48}$$

MCMC methods stochastically explore $\pi(\theta)$ by simulating and representing a set of samples in the form of a Markov Chain (MC). The Markovian nature of the chain means that the current sample only depends on the previous: they are *memoryless*. The key quantity summarising a MC is the transition operator $T(\theta, \theta')$ which describes the probability of moving from θ to θ'. The validity of the MC depends on $T(\theta, \theta')$ being *irreducible* such that a point in the parameter space must be able to reach any other point in the space in a finite number of steps. The chain should also be *aperiodic*, meaning periodic behaviour should not be present. The time until the MC has converged and reached the stationary distribution of $\pi(\theta)$ is dependent on its initial starting point in the parameter space. Therefore it is common practice to discard the initial portion of the chain as *burn-in*. Ensuring convergence to the correct $\pi(\theta)$, the chain is ergodic. MCMC algorithms must

obey detailed balance such that $\pi(\theta)T(\theta,\theta') = \pi(\theta')T(\theta',\theta)$. Three popular methods are described in the subsequent sections.

3.3.1 Metropolis-Hastings Random Walk

The well-studied MHRW algorithm [81, 82, 83] splits $T(\theta,\theta')$ into two distinct steps: the proposal step and the accept/reject step. Firstly, a set of parameters θ' are drawn from a proposal distribution $q(\theta|\theta')$. The proposal is typically assumed to be a Gaussian distribution centered around the current position, θ, with variance, or step size, selected by the user. Employing a small step-size can result in a proposal of θ' that is too close to θ giving rise to high autocorrelation within the MC. If the step-size is too big, θ' may not be within $\pi(\theta)$ resulting in a low acceptance rate which can affect the efficiency of the sampler. The MHRW sampler can also suffer from the *curse of dimensionality*. As the dimensionality of θ increases, it becomes much harder to find good approximations of $\pi(\theta)$.

Secondly, an acceptance step corrects for the fact that θ' may not have been sampled from $\pi(\theta)$. The acceptance probability, $\alpha(\theta,\theta')$, is given by

$$\alpha(\theta,\theta') = \min\left\{1, \frac{\pi(\theta')q(\theta|\theta')}{\pi(\theta)q(\theta'|\theta)}\right\}. \tag{3.49}$$

If the proposal is symmetric, as in the Gaussian case, $q(\theta|\theta') = q(\theta'|\theta)$ and the acceptance probability simplifies to

$$\alpha(\theta,\theta') = \min\left\{1, \frac{\pi(\theta')}{\pi(\theta)}\right\}. \tag{3.50}$$

A random variable u_t is drawn from a Uniform distribution on $[0,1]$. If $u_t \leq \alpha(\theta,\theta')$ then θ' is accepted as part of the MC. If $u_t > \alpha(\theta,\theta')$, θ' is rejected and the MC reverts back to θ. This process is repeated for M MCMC iterations and is outlined in Algorithm 2.

3.3.2 Hamiltonian Monte Carlo

HMC is a gradient based algorithm which uses Hamilton's equations to generate new proposals. Hamilton's equations are a pair of differential equations that describe a system in terms of its position and momentum where the potential of the system is defined by $U(\theta) = -\log(\pi(\theta))$. HMC introduces a momentum vector r which moves samples at θ on a trajectory to θ'. The total energy or Hamiltonian of a system can be expressed as $H(\theta,r) = U(\theta) + K(r)$, and is comprised of the sum of the Kinetic energy $K(r)$, which is

Algorithm 2: Metropolis-Hastings Random Walk

```
Initialise: θ_0
for m = 1, ..., M do
    Sample θ' from q(θ'|θ).
    Compute acceptance probability, α(θ, θ'), using (3.50)
    Sample u_t from Uniform[0, 1]
    if u_t < α(θ, θ') then
        θ = θ'
    else
        θ = θ
    end
end
```

independent of θ, where in the parameter space the samples are, and the potential energy $U(\theta)$, which is independent on the momentum r.

Hamilton's equations describe how the system evolves as a function of time and are:

$$\frac{d\theta}{dt} = \frac{\partial H(\theta, r)}{\partial r}, \frac{dr}{dt} = -\frac{\partial H(\theta, r)}{\partial \theta}. \tag{3.51}$$

The joint density is

$$\begin{aligned} p(\theta, r) \propto \exp(-H(\theta, r)) &= \exp(-U(\theta)) \cdot \exp(-K(r)) \\ &= p(\theta)p(r), \end{aligned} \tag{3.52}$$

therefore θ and r are independent samples from the joint density so r can be sampled from any distribution. For simplicity a Gaussian is often used, and we make that choice here.

Many numerical integration methods exist which discretise Hamilton's equations and can be seen in [84] with the leapfrog method being the go-to method for HMC. Leapfrog is a symplectic method which means the Hamiltonian remains close to its initial value, though not equal to it exactly, as the system is simulated. This means samples are generated with a high accept/reject ratio so the target is explored efficiently. The acceptance step is defined to be

$$\alpha(\theta, \theta') = \min\left\{1, \frac{\exp\left\{\mathcal{L}(\theta') - \frac{1}{2}r' \cdot r'\right\}}{\exp\left\{\mathcal{L}\left(\theta_0\right) - \frac{1}{2}r_0 \cdot r_0\right\}}\right\}. \tag{3.53}$$

As HMC is an MCMC method, it needs to be time-reversible to maintain detailed balance. By negating $t \to -t$ and $r \to -r$ in (3.53), Hamilton's equations are defined to be

$$\frac{d\theta}{d-t} = -\frac{\partial H(\theta, r)}{\partial r}, \frac{d-r}{d-t} = -\frac{\partial H(\theta, r)}{\partial \theta}. \tag{3.54}$$

The form of the equations in (3.52) and (3.54) does not change. In other words, by simulating a leapfrog trajectory with some momentum r, θ moves to θ'. If the same momentum was negated to $-r$, θ' would travel on the same trajectory to θ. The leapfrog step is also volume-preserving. Applying the deterministic transformation $(\theta, r) \to (\theta', r')$ guarantees that the samples are still from the same joint distribution. As outlined in [24], if the new states were proposed with some non-Hamiltonian dynamics, the determinant of the Jacobian matrix for the dynamics would need to be computed. However, the transformation in HMC is both invertible and differentiable so the determinant of a voluming-preserving transformation is one.

Finally, leapfrog is a low-order method which uses relatively few gradient evaluations per step and is therefore computationally cheap. Note there are higher-order methods that use more gradient evaluations per step to take many fewer steps which has been argued that results in a lower total computational cost.

The samples generated are governed by a predetermined number of steps L of size ϵ, decided by the user. HMC is highly sensitive to the choice of these parameters, particularly L. If too large, computation time can be wasted as the trajectory can end close to where it started and, if too small, the proposal can exhibit random-walk behaviour. An example when sampling from a 2-dimensional Gaussian distribution can be seen in Figure 3.7. Figure 3.7(a) and (b) outline how the choice of step-size and parameter L, respectively, can have an impact on the efficiency of the sampler. In some cases it has been shown that randomising L can be beneficial to avoid periodicities in the underlying Hamiltonian dynamics [85].

Algorithm 3: Leapfrog

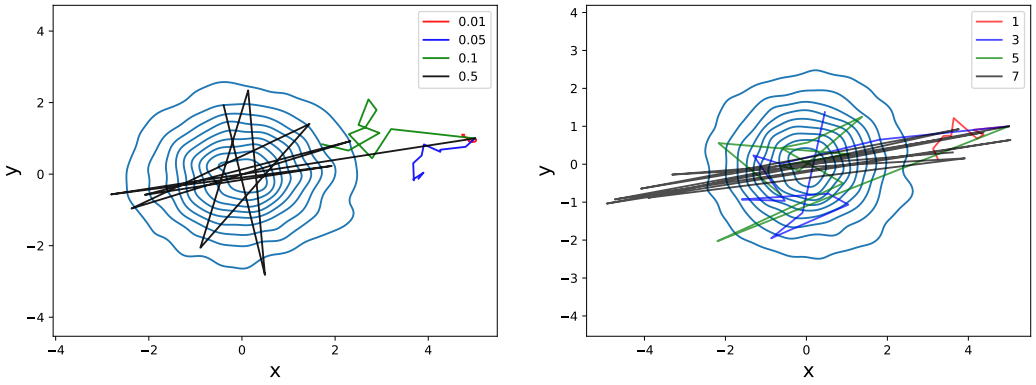
Input: θ, r, ϵ

$r' = r + 0.5 \cdot \epsilon \cdot \nabla \mathcal{L}(\theta)$

$\theta' = \theta + \epsilon \cdot r'$

$r' = r' + 0.5 \cdot \epsilon \cdot \nabla \mathcal{L}(\theta)'$

return θ', r'



(a) (b)

Figure 3.7: Sampling a 2-dimensional Gaussian distribution when changing the step-size of HMC (a) and parameter L (b) for 10 MCMC iterations.

Algorithm 4: Hamiltonian Monte Carlo

Initialise: θ, ϵ, L, M
for $m = 1, \ldots, M$ **do**
  Sample $r \sim \mathcal{N}(0, I)$.
  Set $\theta' \leftarrow \theta$, $r' \leftarrow r$
  for $i = 1, \ldots, L$ **do**
    $\theta', r' =$ Leapfrog(θ', r', ϵ) using Algorithm 3
  end
  Compute acceptance probability, $\alpha(\theta, \theta')$, using (3.53)
  Sample u_t from Uniform$[0, 1]$
  if $u_t < \alpha(\theta, \theta')$ **then**
    $\theta = \theta'$
    $r = r'$
  else
    $\theta = \theta$
    $r = r$
  end
end

3.3.3 No-U-Turn Sampler

NUTS is an extension of HMC which adaptively finds a good number of leapfrog steps to take, for a given step size, and therefore eliminates the need to tune this number. NUTS does this by sampling from a trajectory of generated states in such a way that detailed balance holds. Section 3.3.3.1 outlines how this trajectory is built while Section 3.3.3.2 describes the stopping criteria. Sections 3.3.3.3 and 3.3.3.4 outline how to sample from the generated trajectory and why detailed balance holds, respectively. Further details are presented in Section 3.3.3.5

3.3.3.1 Generating a Trajectory

NUTS generates samples both forwards and backwards in time from the initial state θ^0 using leapfrog, by first sampling an initial momentum m^0. To ensure reversibility, a Bernoulli trial with probability 0.5 is undertaken to pick an initial direction d (either forwards $+\epsilon$, or backwards $-\epsilon$) after which a leapfrog step is taken. Until a U-turn is detected, a new direction is then sampled after every 2^j leapfrog steps where j, initialized to zero, is incremented by one after selecting a new direction. In doing so, NUTS builds a full binary tree of height j where the number of steps taken between tests for a U-turn is double the amount taken since the previous such test and each leaf node represents states along the trajectory.

3.3.3.2 Testing for U-turns

The doubling process described above stops when the position and momentum states at the far left (θ^-, r^-) and far right (θ^+, r^+) of the tree satisfy either of the following conditions:

$$\left(\theta^+ - \theta^-\right) \cdot r^- < 0 \quad \text{or} \quad \left(\theta^+ - \theta^-\right) \cdot r^+ < 0. \tag{3.55}$$

These conditions are met when the trajectory begins to double back on itself, i.e. begins to U-turn. When building a trajectory, once one of these conditions is satisfied the current subtree being built is discarded, and a sample from the resulting full tree (without the discarded subtree) is taken. This process of discarding means that any state in the tree can move to any other state without breaching the U-turn: This is important with respect to ensuring that detailed balance is maintained.

3.3.3.3 Drawing a Sample from the Trajectory

In the original description of NUTS, slice sampling was undertaken to sample a new state from the trajectory. This was done to account for the fact that by using leapfrog, the Hamiltonian dynamics are approximated by numerical integration. Practically, an auxiliary variable, u, is introduced which is sampled uniformly between 0 and $H(\theta^0, r^0)$. Once a U-turn is detected (and the relevant subtree has been discarded) a sample is then uniformly selected from the states which satisfy: $H(\theta, r) > u$. This effectively prevents samples being selected for which the dynamics were poorly modelled (and therefore would have had a poor acceptance probability had HMC been used).

3.3.3.4 Pertinent Elements of the Proof

In following the steps above, the algorithm satisfies detailed balance and is therefore a valid MCMC proposal. For a complete description, see Section 3.1.1 in [26]. The the more pertinent points are outlined here.

For reversibility, a pair of states (θ, r) and (θ', r') must be able to transition to each other. A subset of states are required that could be transitioned to $\mathcal{B}$, where $\mathcal{B}$ is a subset of a potentially larger set of all states observed $\mathcal{C}$, to satisfy:

$$p(\mathcal{B}, \mathcal{C}|\theta, r) = p(\mathcal{B}, \mathcal{C}|\theta', r') \tag{3.56}$$

This is true if the states in NUTS are generated deterministically, i.e. through doubling the number of states. In this instance there will always be a series of directions that will produce the required tree. Furthermore, by discarding the samples in the subtree being generated which contain a U-turn, the possibility of extending the sequence of states is removed such that it contains pairs of states for which the transition is possible from one to the other but not in the other direction.

3.3.3.5 Further Details

The numerical integrator used to simulate the Hamiltonian dynamics is chosen to preserve the geometric structure of the dynamics, i.e. has a unit determinant. NUTS uses the leapfrog method, like HMC, which has this property. As a result, NUTS avoids the situation where, due to numerical errors, some states in the trajectory are unlikely to be sampled.

To avoid excessively large trees, which can result from choosing too small a step size,

it is necessary to upper bound the tree depth. Similarly to HMC, were NUTS able to run with an exact integrator, all the states would be equally likely to be sampled and computing of determinants related to the Jacobian is not required.

3.4 Particle - Markov Chain Monte Carlo

Particle-MCMC combines MCMC and a particle filter, two Monte Carlo methods that use repeated sampling techniques to obtain numerical estimates related to a target distribution $\pi(\theta)$. Exact inference, using p-MCMC, of this target distribution is assumed to be intractable. The original contribution of [29] uses a particle filter to calculate an unbiased estimate of the marginal log-likelihood, $p(y_{1:T}|\theta)$. The log-likelihood can be approximated by summing the unnormalised particle filter weights in (3.41) for $t = 1, .., T$. This is a byproduct of running the particle filter and so no additional calculations are required. The resulting log-likelihood estimate can be used within the MHRW algorithm and allows the acceptance probability to be formulated as

$$\alpha(\theta, \theta') = \min\left\{1, \frac{p\left(\theta'\right) p(y_{1:T}|\theta')q(\theta|\theta')}{p(\theta)p(y_{1:T}|\theta)q(\theta'|\theta)}\right\}, \tag{3.57}$$

where $p(\theta)$ is the prior density and $q(\theta'|\theta)$ the proposal. Algorithms 2 and 5 differ only in their definition of the acceptance probability. Reference [29] prove that the Markov Chain converges to the target distribution $p(\theta|y)$ when using a fixed number of N particles to estimate $p(y_{1:T}|\theta')$. Use of the MHRW proposal in p-MCMC will inherit the same problems as described above in the context of MCMC. To make use of more sophisticated proposals, the gradient of the log posterior of the parameter, θ, must be estimated. This forms the basis of Chapter 4.

3.4.1 Application to SIR Model

An example of inferring the parameters of the discrete SIR model outlined in Section 3.1.4.2 using p-MCMC with the MHRW proposal is presented below. The trace plots for the parameters β (top) and γ (bottom) are presented in Figure 3.8. Four independent Markov chains with different initial starting points were run for 550 MCMC iterations, with the first 50 discarded as *burn-in*. Figures 3.8a and c present the entire Markov chain, while figures 3.8b and 3.8d depict the first 60 iterations. The black vertical line shows the point at which *burn-in* ends. The black horizontal line shows the true parameter value. By the

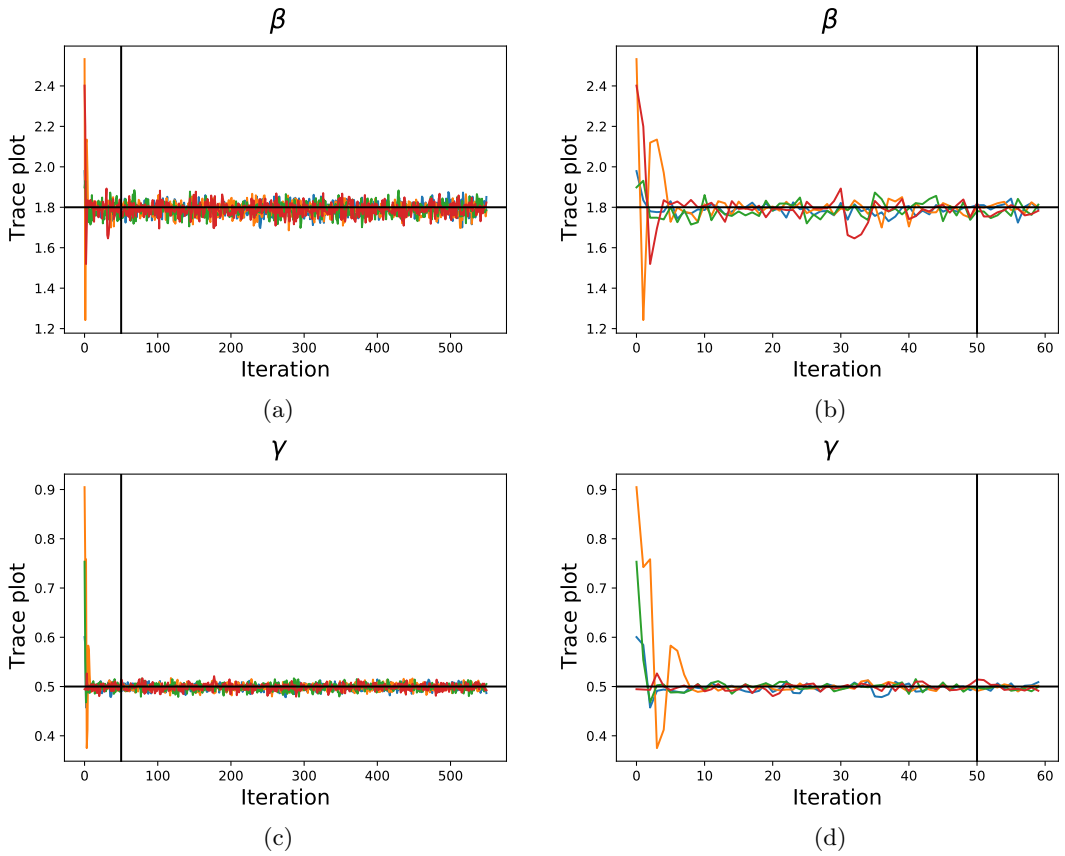


Figure 3.8: Trace plots of β and γ for 550 iterations (left) and the first 60 iterations (right). The vertical black line indicates where *burn-in* ends at 50 iterations.

clear convergence of the Markov chains to the true value in all cases, it is evident that the sampler is able to correctly recover the parameters of the SIR model.

3.5 Evaluating a Markov Chain

The sampling methods described above produce a Markov chain. Understanding how well and how efficiently a Markov chain is sampling is crucial when making comparisons between samplers. In the subsequent sections, the statistics for evaluating Markov chains used throughout this thesis are provided. These metrics also give an indication of when to terminate the sampling process. The ppls mentioned in Section 3.3.3 automatically provide information regarding the effectiveness of the sampler. Three of these methods are

Algorithm 5: Particle- MCMC

Initialise: θ_0, $y_{1:T}$
Run Algorithm 1 with θ_0 to obtain estimate of the log-likelihood, $\log p(y_{1:T}|\theta)$, calculated by (3.41)
for *m = 1, . . . , M* **do**
    Sample θ' from $q(\theta'|\theta)$.
    Run Algorithm 1 with θ' to obtain estimates of the log-likelihood, $\log p(y_{1:T}|\theta')$, calculated by (3.41)
    Compute acceptance probability, $\alpha(\theta, \theta')$, using (3.57)
    Sample u_t from Uniform$[0, 1]$
    if $u_t < \alpha(\theta, \theta')$ **then**
        $\theta = \theta'$
        $\log p(y_{1:T}|\theta)$=$\log p(y_{1:T}|\theta')$
    else
        $\theta = \theta$
        $\log p(y_{1:T}|\theta)$=$\log p(y_{1:T}|\theta)$
    end
end

outlined below.

3.5.1 Effective Sample Size

The effective sample size (ESS) is an explicit approximation of the number of independent samples it would take for the Markov chain to have the same estimation power as the set of auto-correlated samples. Higher ESS values are desirable. The MCMC samplers described in Section 3.3 differ with respect to run time. MHRW runs in less computation time than HMC and NUTS because it does not make gradient evaluations. Depending on the trajectory size, NUTS can make more gradient evaluations per MCMC iteration when compared with HMC. Therefore, calculating the ESS per second from the time taken for the sampler to finish running gives an indication of the efficiency of the sampler.

3.5.2 IACT

The integrated auto-correlated time (IACT) estimates the number of samples, on average, it takes to draw an independent sample from a continuous Markov chain. This is also known as *mixing*, with lower values more desirable.

3.5.3 Gelman Rubin

The Gelman-Rubin diagnostic [86] is a numerical method that determines if multiple chains have converged by comparing the variances between chains. This diagnostic is commonly used in ppls (where it is referred to as $\hat{R}$) to ascertain if the sampler has correctly sampled from the posterior. Stan's documentation states that an $\hat{R}$ value below 1.05 passes their internal diagnostic check.

3.6 Summary

This chapter has outlined the main topics utilised throughout this thesis. Descriptions and examples of SSMs and epidemiological models are provided in Sections 3.1 and 3.1.4, respectively. An overview of the modelling and calibration of these SSMs with particle filters and MCMC methods is presented in Sections 3.2 and 3.3, respectively. Finally, methods for evaluating the Markov chains produced by the samplers are outlined in Section 3.5.

Chapter 4

Efficient Learning of the Parameters of Non-Linear Models using Differentiable Resampling in Particle Filters

It has been widely documented that the sampling and resampling steps in particle filters cannot be differentiated. The *reparameterisation trick* was introduced to allow the sampling step to be reformulated into a differentiable function. In this chapter, the *reparameterisation trick* is extended to include the stochastic input to resampling therefore limiting the discontinuities in the gradient calculation after this step. Knowing the gradients of the prior and likelihood distributions allows for the use of p-MCMC with NUTS as the proposal when estimating parameters.

In this chapter, MALA, HMC with varying numbers of steps and NUTS are compared. Considering three state-space models, it is shown that NUTS improves the mixing of the Markov chain and can produce more accurate results in less computation time.

4.1 Introduction

This chapter aims to complement recent machine learning literature that addresses the problem of differentiatiable resampling.

Use of the *reparameterisation trick* for resampling is described in [87], where the random number seed is fixed in every simulation to produce common random numbers (CRN). The core contribution of this chapter is to fix the random numbers used within the resampling step such that the input to resampling can be conditioned. It is then possible to utilise the estimated gradients within the framework of p-MCMC and to calculate gradient-based proposals for θ. More specifically, this allows use of NUTS as the proposal. A further novel contribution is the provision of full Bayesian parameter estimates (including variances). This differs from existing literature on differentiable particle filtering in the neural network community which focuses exclusively on point-estimates of parameters [39, 40, 41, 43]. The performance of NUTS is also compared with that achieved by HMC and MALA.

An outline of the rest of the chapter is as follows: calculation of the likelihood and gradients is described in Section 4.2.2, methods for propagating the derivatives of the particle weights in Section 4.3 and an extension of the *reparameterisation trick* that includes the stochastic input to resampling in Section 4.4. The likelihood and gradient estimates are tested, particle-HMC (p-HMC) and particle-NUTS (p-NUTS) explained and comparative numerical results detailed in the context of three applications in Section 4.7. Concluding remarks are presented in Section 4.8.

4.2 Particle Filter

The processes and notation in the subsequent sections are the same as those used when describing operations relating to the particle filter outlined in Section 3.2. Calculation of an unbiased estimate of the likelihood, which is used within p-MCMC with an MH proposal (see Section 3.4), is outlined in Section 4.2.1. Gradient based proposals require the gradient of the likelihood w.r.t θ. Section 4.2.2 outlines how to obtain these gradients from a particle filter using the reparameterisation trick.

4.2.1 Calculating the Likelihood

For $t = 1, .., T$ an unbiased estimate of the log-likelihood can be estimated recursively by summing the log-weights in (3.41):

$$p\left(y_{1:T}|\theta\right) \approx \frac{1}{N}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)}. \tag{4.1}$$

This is a byproduct of running the particle filter that is outlined in Section 3.2, therefore additional calculations are not required.

4.2.2 Calculating the Gradient of the Likelihood

Following (4.1), an approximation to the gradient of the likelihood is given by a function of the derivative of the weights[1]:

$$\frac{d}{d\theta}p(y_{1:t}|\theta) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{d}{d\theta}w_{1:t}^{(\theta,i)}. \tag{4.2}$$

For numerical stability, it is typically preferable to propagate values in logs. Applying the chain rule to (3.41) and (4.2) gives

$$\frac{d}{d\theta}\log p(y_{1:t}|\theta) \approx \frac{1}{N}\frac{1}{p(y_{1:t}|\theta)}\sum_{i=1}^{N} w_{1:t}^{(\theta,i)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,i)} \tag{4.3}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\tilde{w}_{1:t}^{(\theta,i)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,i)}. \tag{4.4}$$

Note that in (4.4), the weights $\tilde{w}_{1:t}^{(\theta,i)}$ are normalised, while the log weights are not. The log weights can be calculated recursively by

$$\log w_{1:t}^{(\theta,i)} = \log w_{1:t-1}^{(\theta,i)} + \log\sigma\left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta\right), \tag{4.5}$$

such that

$$\frac{d}{d\theta}\log w_{1:t}^{(\theta,i)} = \frac{d}{d\theta}\log w_{1:t-1}^{(\theta,i)} + \frac{d}{d\theta}\log\sigma\left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta\right), \tag{4.6}$$

where

$$\begin{aligned}\frac{d}{d\theta}\log\sigma\left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta\right) =& \frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, \theta\right) + \frac{d}{d\theta}\log p\left(y_t|x_t^{(\theta,i)}\right) \\ &- \frac{d}{d\theta}\log q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, \theta, y_t\right).\end{aligned} \tag{4.7}$$

[1]Note that this approach differs from that advocated in [52, 53, 54], where a fixed-lag filter (with a user-specified lag) is used to approximate the derivatives. In contrast, in this chapter, the derivatives of the approximation of the likelihood are explicitly calculated.

Therefore, if the single measurement likelihood, $\frac{d}{d\theta}\log p\left(y_t|x_t^{(\theta,i)}\right)$, transition model, $\frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)$, and proposal, $\frac{d}{d\theta}\log q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\right)$, can be differentiated, (an approximation of) the derivative of the log-likelihood for the subsequent timestep can be calculated. Thus, the log-likelihood derivatives are approximated recursively at each timestep. There are, however, a number of challenges involved in this procedure.

If the particle filter uses the transition model as the proposal (as in (3.36)), the likelihood in the weight update does not explicitly depend on θ. Equation (4.7) can therefore be rewritten as

$$\frac{d}{d\theta}\log\sigma\left(x_t^{(\theta,i)},x_{t-1}^{(\theta,i)},\theta\right)=\frac{d}{d\theta}\log p\left(y_t|x_t^{(\theta,i)}\right). \tag{4.8}$$

As such, one may initially suppose that $d\log\sigma/d\theta=0$. If this were true, an induction argument using (4.6) would show that the weight derivatives are always zero and therefore give an approximation of zero for the gradient of the likelihood w.r.t θ. Intuitively incorrect, the flaw in this reasoning is that the likelihood (somewhat implicitly) does, in fact, depend on θ, since $x_t^{(\theta,i)}$ is dependent on θ: application of the chain rule to the remaining term in (4.7) gives

$$\frac{d}{d\theta}\log p\left(y_t|x_t^{(\theta,i)}\right)=\left.\frac{d}{dx}\log p\left(y_t|x\right)\right|_{x=x_t^{(\theta,i)}}\frac{d}{d\theta}x_t^{(\theta,i)}. \tag{4.9}$$

Additionally, $x_t^{(\theta,i)}$ is a random variable sampled from the proposal. To overcome this, the *reparameterisation trick* [42] can be used. Consider the derivative of the likelihood for a fixed random number seed. More precisely, let $\epsilon_{1:t}^{(i)}$ be the vector of $\mathcal{N}(0,1)$ random variables used to sample from the proposal, such that, if $\epsilon_t^{(i)}$ is known, then $x_t^{(\theta,i)}$ is a differentiable deterministic function of $x_{t-1}^{(\theta,i)}$. Under this consideration, the derivative of the log-likelihood in (4.9) can be approximated in following way:

$$\begin{aligned}\frac{d}{d\theta}p\left(y|\theta\right)=&\frac{d}{d\theta}\int p\left(y,\epsilon_{1:t}|\theta\right)d\epsilon_{1:t} &(4.10)\\ =&\int\frac{d}{d\theta}p\left(y,\epsilon_{1:t}|\theta\right)d\epsilon_{1:t} &(4.11)\\ \approx&\frac{1}{N}\sum_{i=1}^{N}\frac{d}{d\theta}p\left(y|\epsilon_{1:t}^{(i)},\theta\right), &(4.12)\end{aligned}$$

where $\epsilon_{1:t}^{(i)} \sim p(\epsilon_{1:t})$ is fixed and, crucially, the derivatives in (4.12) can be calculated.

As a simple example, consider sampling from the dynamics of a random walk proposal and let θ be the standard deviation of the process noise, such that

$$x_t^{(\theta,i)} = x_{t-1}^{(\theta,i)} + \theta\epsilon_t^{(i)} \tag{4.13}$$

and

$$\frac{d}{d\theta}x_t^{(\theta,i)} = \frac{d}{d\theta}x_{t-1}^{(\theta,i)} + \epsilon_t^{(i)}. \tag{4.14}$$

Then, (4.14) can be calculated recursively and used to derive (4.9).

More generally, to calculate the non-zero derivatives of the weights, the particle derivatives $dx_t^{(\theta,i)}/d\theta$ must be propagated. This is discussed in the following section.

4.3 Calculating the Derivatives

In order to propagate the derivatives of the particle weights in (4.6) and (4.7) the following must be calculated:

- The derivatives of the particle states

$$\frac{dx_t^{(\theta,i)}}{d\theta} \tag{4.15}$$

- The derivatives of the proposal log pdfs

$$\frac{d}{d\theta}\log q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\right) \tag{4.16}$$

- The derivatives of the transition log pdfs

$$\frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right) \tag{4.17}$$

- The derivatives of the single measurement likelihood log pdfs

$$\frac{d}{d\theta}\log p\left(y_t|x_{t-1}^{(\theta,i)}\right). \tag{4.18}$$

Section 4.3.1 shows how to calculate these derivatives in turn.

4.3.1 Derivative of the Particle States

To calculate $dx_{t-1}^{(\theta,i)}/d\theta$, suppose the proposal takes the form

$$q\Big(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\Big)=\mathcal{N}\left(x_t^{(\theta,i)};\mu\left(x_{t-1}^{(\theta,i)},\theta,y_t\right),C\left(x_{t-1}^{(\theta,i)},\theta,y_t\right)\right), \tag{4.19}$$

where $\mu(\cdot)$ and $C(\cdot)$ are functions of the old particle state, the parameter and the observation. Such a generic description can articulate sampling from the prior, or defining a proposal using a Kalman filter with the predicted mean and covariance given by the motion model.

If the proposal noise $\epsilon_t^{(i)} \sim \mathcal{N}(\cdot;0,I_{n_X})$ is sampled in advance, the new particle states can be written as a deterministic function:

$$\begin{aligned} x_t^{(\theta,i)} &= f(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}) \\ &\triangleq \mu(x_{t-1}^{(\theta,i)},\theta,y_t)+\sqrt{C(x_{t-1}^{(\theta,i)},\theta,y_t)}\times\epsilon_t^{(i)}. \end{aligned} \tag{4.20}$$

Care must be taken when computing the derivative of (4.20) w.r.t. θ since $x_{t-1}^{(\theta,i)}$ is also a function of θ, with different particles $x_{t-1}^{(\theta,i)}$ sampled for different values of θ. By the chain rule, the derivative is given by

$$\frac{dx_t^{(\theta,i)}}{d\theta} = \frac{d}{d\theta}f(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}) \tag{4.21}$$

$$= \frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}\frac{d\theta}{d\theta} \tag{4.22}$$

$$= \frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}. \tag{4.23}$$

Note that, the derivative $df/d\theta$ in (4.21) and the partial derivative $\partial f/\partial\theta$ in (4.22) are not equivalent — see Appendix A.1 for the distinction. In addition, the terms are generally matrix-valued: $\partial f/\partial x_{t-1}^i$ is an $n_X \times n_X$ matrix, and $dx_{t-1}^i/d\theta$ and $\partial f/\partial\theta$ are $n_X \times n_\Theta$ matrices.

4.3.2 Derivative of the Proposal

The proposal log pdf can be written as

$$\log q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta,y_t\right) = Q\left(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}\right), \tag{4.24}$$

where, under the assumption of a Gaussian proposal and dropping the fixed values $\epsilon_t^{(i)}$ and y_t for notational convenience,

$$\begin{aligned} Q\left(x_{t-1}^{(\theta,i)},\theta\right) &\triangleq \log q\left(f(x_{t-1}^{(\theta,i)},\theta)|x_{t-1}^{(\theta,i)}\right) && (4.25)\\ &= \log\mathcal{N}\left(f(x_{t-1}^{(\theta,i)},\theta);\mu(x_{t-1}^{(\theta,i)},\theta),C(x_{t-1}^{(\theta,i)},\theta)\right). && (4.26)\end{aligned}$$

Then, the derivative of the proposal log pdf is derived by

$$\begin{aligned} \frac{d}{d\theta}Q(x_{t-1}^{(\theta,i)},\theta) =& \frac{\partial}{\partial f}\log\mathcal{N}(f;\mu,C)\left(\frac{df}{d\theta}+\frac{d\mu}{d\theta}+\frac{dC}{d\theta}\right) && (4.27)\\ =& \frac{\partial}{\partial f}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}\right)\\ &+\frac{\partial}{\partial\mu}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial\mu}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial\mu}{\partial\theta}\right)\\ &+\frac{\partial}{\partial C}\log\mathcal{N}(f;\mu,C)\left(\frac{\partial C}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial C}{\partial\theta}\right), && (4.28)\end{aligned}$$

where $\mu=\mu(x_{t-1}^{(\theta,i)},\theta)$ and $C=C(x_{t-1}^{(\theta,i)},\theta)$. The derivatives of $\log\mathcal{N}(f;\mu,C)$ are provided in Appendix A.3.

4.3.3 Derivative of the Prior

To calculate $\frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)$, let

$$\begin{aligned} P(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}) &\triangleq \log p\left(f\left(x_{t-1}^{(\theta,i)},\theta,y_t,\epsilon_t^{(i)}\right)|x_{t-1}^{(\theta,i)},\theta\right) && (4.29)\\ &= \log\mathcal{N}\left(f\left(x_{t-1}^{(\theta,i)}\right);a\left(x_{t-1}^{(\theta,i)},\theta\right),\Sigma(\theta)\right), && (4.30)\end{aligned}$$

where the transition model is assumed to have additive Gaussian noise that is independent of $x_{t-1}^{(\theta,i)}$. Then,

$$\begin{aligned}\frac{d}{d\theta}P(x_{t-1}^{(\theta,i)},\theta) =& \frac{\partial}{\partial f}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial f}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial f}{\partial\theta}\right)\\ &+\frac{\partial}{\partial a}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial a}{\partial x_{t-1}^{(\theta,i)}}\frac{dx_{t-1}^{(\theta,i)}}{d\theta}+\frac{\partial a}{\partial\theta}\right)\\ &+\frac{\partial}{\partial\Sigma}\log\mathcal{N}(f;a,\Sigma)\left(\frac{\partial\Sigma}{\partial\theta}\right).\end{aligned} \tag{4.31}$$

Note that, the derivatives of $\log\mathcal{N}(f;a,\Sigma)$ are evaluated at $a(x_{t-1}^{(\theta,i)})$ (the prior mean) and not at μ (the proposal mean), as was the case in (4.28).

4.3.4 Derivative of the Likelihood

To calculate $\frac{d}{d\theta}\log p\left(y_t|x_t^{(\theta,i)}\right)$, let

$$L(x_t^{(\theta,i)},\theta,y_t) \triangleq \log p\left(y_t|x_t^{(\theta,i)}\right), \tag{4.32}$$

where the right-hand side of (4.32) is the log-likelihood in (4.9). Consider the case in which the likelihood is Gaussian with a variance that is independent of $x_t^{(\theta,i)}$, such that

$$L(x_t^{(\theta,i)},\theta,y_t) \triangleq \log\mathcal{N}\left(y_t;h(x_t^{(\theta,i)},\theta),R(\theta)\right). \tag{4.33}$$

Then, by application of the chain rule,

$$\begin{aligned}\frac{d}{d\theta}L\left(x_t^{(\theta,i)},\theta,y_t\right) =& \frac{\partial}{\partial h}\log\mathcal{N}(y_t;h,R)\left(\frac{\partial h}{\partial x_t^{(\theta,i)}}\frac{dx_t^{(\theta,i)}}{d\theta}+\frac{\partial h}{\partial\theta}\right)\\ &+\frac{\partial}{\partial R}\log\mathcal{N}(y_t;h,R)\frac{dR}{d\theta}.\end{aligned} \tag{4.34}$$

4.4 Resampling for a Differentiable Particle Filter

Unlike for the standard particle filter described in Section 3.2, the weight derivatives

$$\frac{d}{d\theta} w_{1:t}^{(\theta,i)} \tag{4.35}$$

and the particle derivatives

$$\frac{d}{d\theta} x_t^{(\theta,i)} \tag{4.36}$$

need to be resampled. The resampling procedure adopted in this chapter is as follows:

Let

$$c_t^{(\theta,i)} = \frac{\sum_{j=1}^{i} w_{1:t}^{(j,\theta)}}{\sum_{j=1}^{N} w_{1:t}^{(j,\theta)}} \tag{4.37}$$

be the normalised cumulative weights and let

$$\kappa_i = \kappa\left(\nu_t^i, w_{1:t}^{1:N}\right) = \sum_{j=0}^{N-1} \left[\nu_t^i > c_t^{(j,\theta)}\right] \tag{4.38}$$

be the index sampled for particle i, where $\nu_t^i \sim \text{Uniform}((0,1])$ are independent for each particle and timestep. The particle indices are sampled according to a categorical distribution, giving a multinomial resampler, where each index is resampled with a probability proportional to its weight. Alternative resampling schemes exist that could reduce the variance of estimates.

The resampled weights are equal, while preserving the original sum:

$$x_t'^{(\theta,i)} = x_t^{(\theta,\kappa_i)}, \tag{4.39}$$

$$w_{1:t}'^{(\theta,i)} = \frac{1}{N} \sum_{j=1}^{N} w_{1:t}^{(\theta,j)}. \tag{4.40}$$

From (4.40), it is clear that

$$\frac{d}{d\theta} w_{1:t}'^{(\theta,i)} = \frac{1}{N} \sum_{j=1}^{N} \frac{d}{d\theta} w_{1:t}^{(\theta,j)}. \tag{4.41}$$

Converting this derivative to the derivative of log weights, application of the chain rule gives

$$\frac{d}{d\theta}\log w_{1:t}^{\prime(\theta,i)} = \frac{1}{N}\frac{1}{w_{1:t}^{\prime(\theta,i)}}\sum_{j=1}^{N} w_{1:t}^{(\theta,j)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,j)} \tag{4.42}$$

$$= \sum_{j=1}^{N}\tilde{w}_{1:t}^{(\theta,j)}\frac{d}{d\theta}\log w_{1:t}^{(\theta,j)}, \tag{4.43}$$

where $\tilde{w}_{1:t}^{(\theta,j)}$ are the normalised weights.

To obtain the particle gradient, note that, for differentiable κ,

$$\frac{d}{d\theta}x_t^{\prime(\theta,i)} = \frac{\partial}{\partial\kappa}x_t\left(\theta,\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right)\right)\frac{\partial}{\partial\theta}\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right) + \frac{d}{d\theta}x_t\left(\theta,\kappa\left(\nu_t^i, w_{1:t}^{1:N}\right)\right). \tag{4.44}$$

Since $\partial\kappa/\partial\theta = 0$, except where

$$\nu_t^i = c_t^{(\theta,j)} \text{ for } i,j = 1,\ldots,N, \tag{4.45}$$

then

$$\frac{d}{d\theta}x_t^{\prime(\theta,i)} = \frac{d}{d\theta}x_t^{(\theta,\kappa_i)} \tag{4.46}$$

almost surely, such that derivative of the resampled state is obtained by taking the derivative of the parent particle.

4.4.1 Discontinuities after a Resampling Realisation

Sampling with CRN results in a deterministic function $f(\theta)$, such that evaluating the function twice gives the same output. Different outputs are obtained when sampling without CRN. When using CRN resampling, a discontinuity occurs in the estimate of the log-likelihood in Figures 4.1 (a) and (c) when two distinct values of θ cause different resampling realisations to occur. On the left of the discontinuity, resampling takes place at the same times for all θ and all particles have the same parents at all resampling events: the particles for different values of θ share a single family tree. On the right of the discontinuity, resampling realisations, and so the family tree, change. Since the approximation of the likelihood (and its gradient) is a function of the family tree, this change results in a

discontinuity in the likelihood approximation.

The best approach to limiting these discontinuities is to use a high-performance proposal. This is exemplified when comparing results obtained using the prior with those obtained using the optimal proposal (which can, in some settings, be derived using a Kalman Filter). Consider the following simple example to demonstrate this point. Suppose that the state is a single real number, with a motion model given by a random walk with zero initial mean and standard deviation, of both the initial state and each subsequent propagation of θ. Given the state and observations are independent of θ and are defined by the target state plus errors of known and fixed variance, R:

$$p(x_1) = \mathcal{N}\left(x_1; 0, \theta^2\right), \tag{4.47}$$

$$p(x_k \mid x_{k-1}, \theta) = \mathcal{N}\left(x_k; x_{k-1}, \theta^2\right), \tag{4.48}$$

$$p(y_k \mid x_k, \theta) = \mathcal{N}(y_k; x_k, R). \tag{4.49}$$

Appendix A.2 outlines how to calculate the mean and covariance of the optimal proposal for each particle x_{t-1}^i, in addition to the necessary derivatives.

Figure 4.1 presents results obtained when running Algorithm 6 and computing an estimate of the log-likelihood and the associated gradient across a range of 500 values of θ, equally spaced from 1 to 4, for $N = 2000$ particles and $T = 250$ observations. The true value is θ=2. The log-likelihood and its gradient w.r.t. θ, at each instance of θ, is presented in Figures 4.1 (a) and (b), respectively. Figures 4.1 (c) and (d) are magnified instances of these plots. Results are shown for the Kalman Filter, two runs of multinomial resampling (to indicate the difference in results when running the simulation twice) and CRN resampling.

Figure 4.1 (a) shows no clear differences in the graphs of the log-likelihood for the estimates given by the Kalman Filter and the estimates produced by the particle filter when using different combinations of the prior and optimal proposal for CRN and multinomial resampling. However, there is a notable difference when comparing the gradient of the log-likelihood w.r.t. θ. Using the prior as the proposal results in large discontinuities in the estimate when compared with the optimal proposal. Figures 4.1 (c) and (d) show that using CRN and the optimal proposal produces piecewise continuous estimates. This is in contrast to multinomial resampling where fluctutations are apparent. Using an optimal proposal, or an approximation to such a proposal, in conjunction with CRN resampling is therefore advocated. A good proposal will minimise the variance of the incremental

weights at the current timestep. The process of selecting a good proposal can be time consuming, however, as outlined in [88], can be critical in obtaining favourable results in other contexts.

Algorithm 6: Differentiable Particle Filter

Input: θ, $y_{1:T}$

Initialise: x_0^i, $\frac{d}{d\theta}x_0^i$, $\log(w_0^i)$, $\frac{d}{d\theta}\log(w_0^i)$

for $t = 1, \ldots, T$ **do**

  Calculate the number of effective samples, N_{eff}, using (3.45).

  If $N_{eff} < N/2$, resample x_{t-1}^i, $\log(w_{1:k-1}^i)$, $dx_{t-1}^i/d\theta$, $d\log(w_{1:t-1}^i)/d\theta$ as described in Section 4.4.

  Sample the new particles x_t^i and calculate the partial derivatives

$$\frac{\partial}{\partial\theta}f(x_{t-1}^i,\theta), \frac{\partial}{dx_{t-1}^i}f(x_{t-1}^i,\theta).$$

  Calculate the proposal mean, $\mu(x_{t-1}^i)$, and covariance, $C(x_{t-1}^i)$, for each particle x_{t-1}^i, in addition to their derivatives

$$\frac{\partial}{\partial\theta}\mu(x_{t-1}^i), \frac{\partial}{x_{t-1}^i}\mu(x_{t-1}^i), \frac{\partial}{\partial\theta}C(x_{t-1}^i), \frac{\partial}{x_{t-1}^i}C(x_{t-1}^i),$$

  as seen in Section 4.3.1.

  Estimate the particle gradients, $\frac{d}{d\theta}x_t^i$, using (4.3.1).

  Estimate the derivatives of the prior, proposal and likelihood using the methods outlined in Sections 4.3.2, 4.3.3 and 4.3.4, respectively.

  Evaluate the new log weights, $\log w_{1:k}^i$, and log weight derivatives, $\frac{d}{d\theta}\log w_{1:t}^i$, using (4.5) and (4.6), respectively.

end

Evaluate the final log-likelihood, $\log p(y_{1:T}|\theta)$, and associated derivative, $\frac{d}{d\theta}\log p(y_{1:T}|\theta)$, using (3.41) and (4.4), respectively.

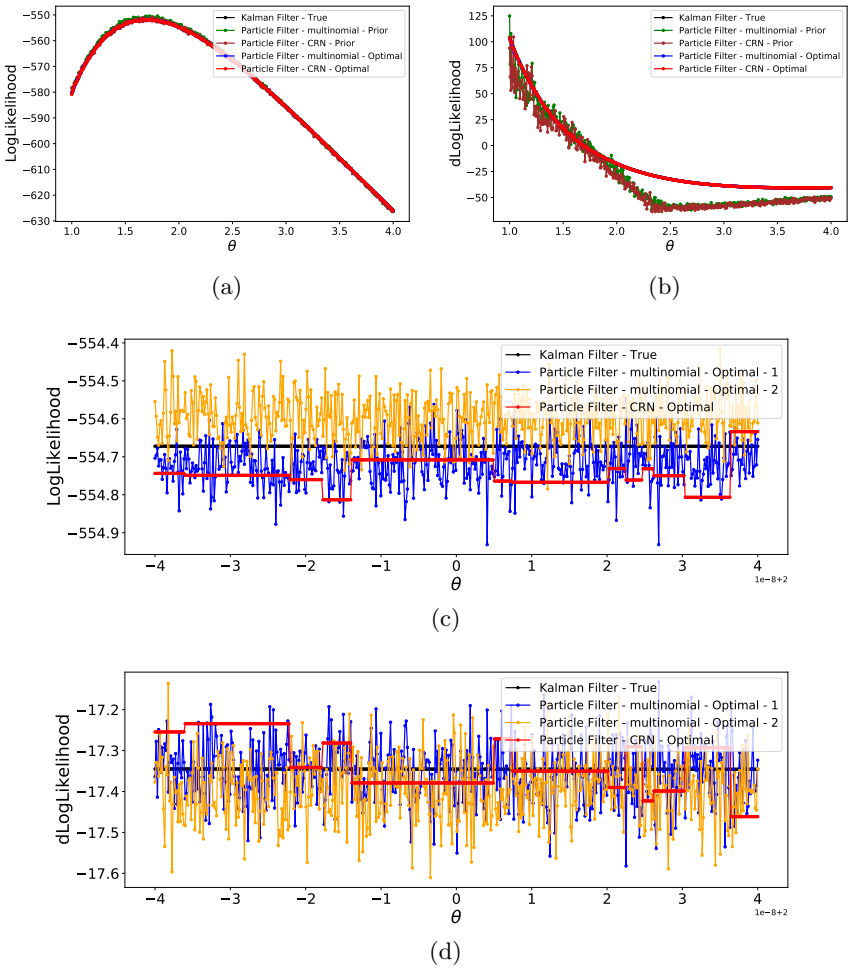


Figure 4.1: Plots of (a) the log-likelihood, (b) the gradient of the log-likelihood w.r.t. θ, (c) a magnified section of the log-likelihood and (d) the associated gradient. All plots: true values given by the Kalman Filter (black), particle filter using CRN resampling (red) and multinomial resampling (blue and orange).

4.5 Differentiable Particle Filters

In the subsequent sections, the differentiable particle filters considered for comparison are described. This comparison includes the novel method of using CRN described in this chapter.

4.5.1 Soft Resampling

Soft resampling, introduced in [44] and utilised in [39], considers a differentiable approximation to resampling which involves drawing from the distribution $q(n) = \alpha w_{1:t}^{(\theta,i)} + (1-\alpha)/N$, with $\alpha \in [0, 1]$ representing a trade-off parameter. If $\alpha = 1$, regular resampling is used and if $\alpha = 0$ the algorithm performs subsampling. The new weights are calculated by

$$w_{1:t}^{\prime(\theta,i)} = \frac{w_{1:t}^{(\theta,i)}}{\alpha w_{1:t}^{(\theta,i)} + (1-\alpha)1/N}. \tag{4.50}$$

This gives non-zero estimates of the gradient because the dependency on the previous weights is maintained. By changing α, this method trades resampling quality for biased gradient estimates.

4.5.2 Gumbel Softmax

The Gumbel-Max trick [89] provides a method for sampling a variable, z, from a categorical distribution containing class probabilities, π_x. If it is assumed that the categorical samples are one-hot vectors, z is sampled by

$$z = \text{onehot}\left(argmax_i \left\{G_i + \log\left(\pi_i\right)\right\}\right) \tag{4.51}$$

where $x_i = G_i + \log(\pi_i)$ and G_i are independently sampled from Gumbel$(0, 1)$. The summation in (4.51) is similar to the reparametrisation trick described previously, however, in this case, the *argmax* function is not differentiable. The work outlined in [45] and [46] describes a differentiable approximation to *argmax*, called *softmax*, defined to be

$$z = \frac{\exp\left(\frac{x_k}{\lambda}\right)}{\sum_{i=1}^{n} \exp\left(\frac{x_i}{\lambda}\right)}. \tag{4.52}$$

The temperature parameter, λ, is defined by the user and controls how closely the resulting Gumbel-softmax distribution approximates the categorical distribution.

4.5.3 Optimal Transport

A fully differentiable particle filter that resamples using optimal transport ideas proposed in [90] is described in [43]. This method ensures unbiased gradient estimates whilst incorporating bias in the log-likelihood estimate. This requires additional hyper-parameters and runs in $O(N^2)$ time complexity. It can therefore be computationally expensive.

4.5.4 Fisher's Identity to Calculate Gradient of the Log-likelihood

As described in Chapter 1, [48] recursively computes the gradient of the log-likelihood using Fisher's Identity. This method is summarised as follows:

$$\begin{aligned}
\frac{d}{d\theta}\log p\left(y_{1:t}|\theta\right) =& \frac{1}{p\left(y_{1:t}|\theta\right)}\frac{d}{d\theta}p\left(y_{1:t}|\theta\right) && (4.53)\\
=& \frac{1}{p\left(y_{1:t}|\theta\right)}\frac{d}{d\theta}\int p\left(y_{1:t},x_{1:t}|\theta\right)dx_{1:t} && (4.54)\\
=& \frac{1}{p\left(y_{1:t}|\theta\right)}\int \frac{d}{d\theta}p\left(y_{1:t},x_{1:t}|\theta\right)dx_{1:t} && (4.55)\\
=& \int \frac{p\left(y_{1:t},x_{1:t}|\theta\right)}{p\left(y_{1:t}|\theta\right)}\frac{d}{d\theta}\log p\left(y_{1:t},x_{1:t}|\theta\right)dx_{1:t} && (4.56)\\
=& \int p\left(x_{1:t}|y_{1:t},\theta\right)\frac{d}{d\theta}\log p\left(x_{1:t},y_{1:t}|\theta\right)dx_{1:t} && (4.57)\\
\approx& \frac{1}{N}\sum_{i=1}^{N}\tilde{w}_{1:t}^{(\theta,i)}\underbrace{\frac{d}{d\theta}\log p\left(x_{1:t}^{(\theta,i)},y_{1:t}|\theta\right)}_{\alpha_n^{(\theta,i)}}. && (4.58)
\end{aligned}$$

The term inside the sum can be calculated recursively:

$$\alpha_n^{(\theta,i)} = \alpha_{n-1}^{(\theta,i)} + \frac{d}{d\theta}\log p\left(y_t|x_{t-1}^{(\theta,i)}\right) + \frac{d}{d\theta}\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right), \tag{4.59}$$

where $\log p\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)},\theta\right)$ is the derivative of the prior and $\frac{d}{d\theta}\log p\left(y_t|x_{t-1}^{(\theta,i)}\right)$ the derivative of the log-likelihood (see (3.34), (4.3.3) and (4.3.4), respectively).

Recent work outlines how to perform this calculation in the framework of PyTorch without having to modify the forward pass [91]. A *stop-gradient* operator that stops the

gradients of the weights flowing into the resampling distribution is included. Much like the method described in this chapter, minimal changes to the particle filtering algorithm need to be employed.

References [48] and [49] focus on unbiased estimators of the gradient of the log-likelihood, using the expectation of the gradient of the logarithm as the approximation. In this chapter, the gradient of the logarithm of the expectation is instead used as the approximation. A feature of the approximation considered here, perhaps surprisingly, is that all samples from all iterations are used, whereas the approach in [48] and [49] recursively calculates along the trajectory of each particle. If, as is likely, the particle trajectories are degenerate, the approach in [48] and [49] will face limitations in using the diversity of states early in the trajectory to inform the gradient estimate. In short, choosing between the two approximations is likely to become a bias-variance trade-off.

4.6 Estimation of Parameters

If there exists a prior, $p(\theta)$, for which the likelihood, $p(y_{1:T}|\theta)$, or log-likelihood[2] can be calculated, then p-MCMC can be used to estimate $p(\theta|y_{1:T}) \propto p(\theta)\, p(y_{1:T}|\theta)$. The gradient of the log-posterior of θ is given by

$$\nabla \log p(\theta|y_{1:t}) = \nabla \log p(\theta) + \nabla \log p(y_{1:t}|\theta), \tag{4.60}$$

where $\nabla \log p(\theta)$ is the gradient of the log-prior and $\nabla \log p(y_{1:t}|\theta)$ the gradient of the log-likelihood. If $\nabla \log p(\theta|y_{1:t})$ is known, it is possible to reduce the correlation between successive sampled states by proposing moves to distant states in $\pi(\theta)$ and maintain a high probability of acceptance.

4.6.1 Hamiltonian Monte Carlo and the No-U-Turn Sampler

For description of the HMC and NUTS[3] algorithms used in the subsequent sections, refer to Sections 3.3.2 and 3.3.3, respectively.

[2]The log-likelihood is likely to be more stable numerically.

[3]Note, using CRN when calculating the gradient from a particle filter results in gradients that are a deterministic function of the parameter. The Jacobian terms therefore cancel and integration over the possible dynamic states is not required, as is the case for regular NUTS.

4.6.2 Metropolis-Adjusted Langevin Algorithm (MALA)

MALA is a MHRW proposal that includes gradient information about the log-posterior (as seen in [54]):

$$\theta' = \mathrm{N}\left(\theta + \frac{1}{2}\Gamma\nabla \log p(\theta|y_{1:T}), \Gamma\right), \tag{4.61}$$

where $\Gamma = \gamma^2 I_d$, for step size γ. In a similar way to [53], the algorithm is executed for varying step size, with the step size that provides an acceptance rate of approximately 0.3 in the stationary phase selected. Although MALA is presented as an independent proposal, it is a special case of HMC when $L = 1$.

Algorithm 7: Particle - HMC or NUTS

Input: θ_0, $y_{1:T}$, M, L
$\ell(\theta)$, $\nabla\mathcal{L}(\theta)$ = Run Algorithm 6
ϵ = Find Reasonable $\epsilon(\theta_0)$
for *i = 1 to M* **do**
    HMC = Algorithm 1 in [26] or
    NUTS = Algorithm 3 in [26]
end
Function `Leapfrog`(θ, r, $y_{1:T}$, $\nabla\mathcal{L}(\theta)$)**:**
    $r' = r + 0.5 \cdot \epsilon \cdot \nabla\mathcal{L}(\theta)$
    $\theta' = \theta + \epsilon \cdot r'$
    $\ell(\theta)$, $\nabla\mathcal{L}(\theta)'$ = Run Algorithm 6
    $r' = r' + 0.5 \cdot \epsilon \cdot \nabla\mathcal{L}(\theta)'$
    return θ', r', $\ell(\theta)'$, $\nabla\mathcal{L}(\theta)'$

4.7 Numerical Experiments

In the subsequent sections, the differentiable particle filters and sampling algorithms (see Sections 4.5 and 4.6, respectively) are evaluated when inferring the parameters in three SSMs. The three examples are the LGSSM, SVM and epidemiological models presented in Sections 3.1.1, 3.1.2 and 3.1.4.1, respectively.

4.7.1 Linear Gaussian State-Space Model

The following example corresponds to that in Section 3.1.1, all parameter notations and priors remain the same.

The "optimal" proposal (3.38) is used within the particle filter and can be derived from the properties of the distributions in (3.3) and (3.4). This gives

$$q\left(x_t|x_{t-1}, y_t\right) = \mathcal{N}\left(x_t; \sigma^2\left[\sigma_e^{-2} y_t + \sigma_v^{-2}\phi x_{t-1}\right], \sigma^2\right), \tag{4.62}$$

where $\sigma^{-2} = \sigma_v^{-2} + \sigma_e^{-2}$. Weights are updated using (3.39) such that

$$w_{1:t}^{(\theta,i)} = \mathcal{N}\left(y_t; \phi x_t, \sigma_v^2 + \sigma_e^2\right) w_{1:t-1}^{(\theta,i)}. \tag{4.63}$$

4.7.1.1 Results

First, the differentiable particle filters described in Section 4.5 are compared in terms of computation time and MSE between true and inferred values of $\theta = \{\phi, \sigma_v\}$ in the LGSSM described in Section 4.7.1. The true values of ϕ and σ_v are 0.7 and 1.2, respectively. NUTS is used as the proposal and the particle filter run with varying numbers of particles, N, and observations, T, over $M = 50$ MCMC iterations. Results are presented in Table 4.1, with the MSE and time taken in seconds averaged over 10 runs (with different random number seeds).

Use of CRN and FI (Fisher's identity) consistently results in the lowest computation time. One reason for this is that minimal changes to the resampling step are introduced when compared to the alternative methods considered in Table 4.1. For $T = 25$ and $T = 100$, CRN results in the lowest MSE when running with $N = 32$, 64 and 128 and $N = 16$, 32, 64, and 128, respectively. Optimal transport resampling results in lower MSE estimates in some experiments, however, the computation time is considerably higher than for CRN and FI.

		T=25					T=50					T=100				
		CRN	SR	GS	OT	FI	CRN	SR	GS	OT	FI	CRN	SR	GS	OT	FI
N=16	MSE	0.151	1.446	0.179	0.141	**0.135**	0.075	0.438	0.076	**0.025**	**0.025**	**0.035**	0.056	0.045	0.040	0.047
	Time (sec)	593	730	554	2062	582	728	1423	765	6256	1590	939	2684	1018	4396	1005
N=32	MSE	**0.112**	0.899	0.174	0.124	0.121	0.077	0.297	0.078	**0.020**	0.021	**0.035**	0.050	0.045	**0.035**	0.036
	Time (sec)	558	816	587	1667	589	790	1478	784	6839	1339	941	2601	1583	5292	1014
N=64	MSE	**0.121**	1.091	0.165	**0.121**	0.126	0.074	0.452	0.084	0.069	**0.025**	**0.025**	0.396	0.050	0.044	0.049
	Time (sec)	552	747	852	1647	608	750	1481	1229	2303	1369	1952	2571	2959	5281	1022
N=128	MSE	**0.129**	1.576	0.135	0.142	0.162	0.078	0.318	0.074	0.082	**0.027**	**0.026**	0.825	0.043	0.044	0.044
	Time (sec)	583	880	1152	1827	582	738	1452	1423	2592	1383	1586	2621	3034	5651	1020

Table 4.1: Average MSE and computation time in seconds for estimation of $\theta = \{\phi, \sigma_v\}$ in the LGSS model for varying N and T, using the differentiable particle filters outlined in Section 4.5: CRN = us, SR = soft resampling, GS = gumbel-softmax, OT = optimal transport and FI = Fisher's identity. Results are averaged over 10 runs using different random number seeds with NUTS as the proposal.

		N = 512			N = 1024		
		T (Secs)	*MSE*	*NGE*	*T (Secs)*	*MSE*	*NGE*
	MALA	4.342	0.306	9	6.288	0.300	9
HMC	L2	3.70	0.36±0.01	18	14.21	0.36±0.02	18
	L4	7.26	0.31±0.02	36	32.34	0.31±0.04	36
	L6	11.39	0.27±0.03	54	57.84	0.26±0.06	54
	L8	18.86	0.25±0.05	72	54.18	0.25±0.07	72
	L10	24.46	0.22±0.06	90	103.80	0.23±0.08	90
	NUTS	35.77	0.23±0.08	106	69.91	0.19±0.07	66

Table 4.2: Computation time in seconds, MSE and the number of gradient evaluations (NGE) for estimation of $\theta = \{\phi, \sigma_v, \sigma_e\}$ in the LGSS model for varying N, using different MCMC proposals. $T = 100$ observations, $M = 10$ MCMC iterations. CRN resampling is used in all experiments. Results are averaged over 10 runs using different random number seeds.

Next, the proposals outlined in Section 4.6 are compared. As explained previously, using NUTS eliminates the need to manually select the length parameter, L, in HMC by adaptively selecting the parameter at every iteration. Therefore, it is likely that NUTS makes more than one target evaluation per iteration and so is more computationally costly than MALA, where one evaluation is made, and HMC for certain values of L. As such, in Table 4.2, the number of gradient evaluations is presented as well as the average MSE between the true and inferred values of $\theta = \{\phi, \sigma_v, \sigma_e\}$ and the computation time for the LGSSM described in Section 4.7.1. The true values of ϕ, σ_v and σ_e are 0.7, 1.2 and 1, respectively. The setup of the experiment is as follows: $T = 100$, $M = 10$ and $N = 512$, 1024. CRN resampling is used in all experiments. Table 4.2 exemplifies the benefit of NUTS over HMC, without the need to optimise the number of steps, L: NUTS obtains a lower MSE in a shorter run time than HMC.

Finally, results for use of NUTS and CRN resampling are presented alongside a commonly used diagnostic for determining whether three independent chains with different initial starting values of $\theta = \{\phi, \sigma_v, \sigma_e\}$ have converged. Consider the initialisation $N = 750$ particles, $T = 250$ observations and $M = 500$ MCMC iterations (first 100 discarded as *burn-in*). The true values of $\theta = \{\phi, \sigma_v, \sigma_e\}$ are $0.7, 1.2$ and 1, respectively. The trace and density plots for the accepted samples of θ are given on the left and right of Figure 4.2 (a), respectively. These plots provide an indication as to how well the chains have converged to their stationary distributions. Figures 4.2(b)-(d) present 1-dimensional histograms, plotted

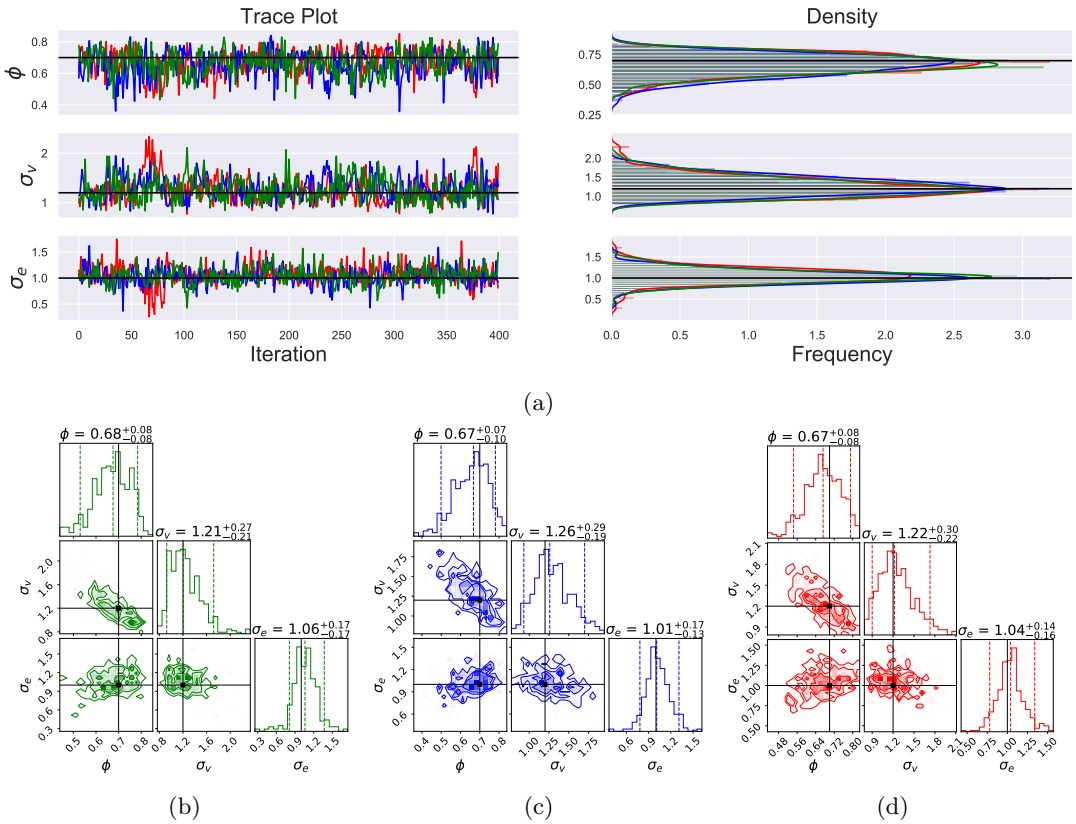


Figure 4.2: (a) Trace plots (left) and density estimates (right) of three independent chains with different initial starting values of θ. Horizontal black lines indicate the true values. (b)-(d) 1-dimensional corner plots for the three independent chains. Horizontal and vertical black lines indicate the true values.

using [92], for the three chains. The uncertainties associated with the estimates of θ are also provided. The mean estimate of θ derived from the three chains is $[0.67, 1.23, 1.04]$ and the corresponding Gelman-Rubin statistics (see Section 3.5.3) 1.0091, 1.007 and 1.0094, respectively.

4.7.2 Stochastic Volatility Model

The following example corresponds to that in Section 3.1.2, all parameter notations and priors remain the same.

The prior is set to be the proposal (3.36), such that the incremental log weight (3.34)

becomes

$$\log \sigma \left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta \right) = \log p \left(y_t | x_t^{(\theta,i)} \right) \tag{4.64}$$

and the associated gradient (see (4.7))

$$\frac{d}{d\theta} \log \sigma \left(x_t^{(\theta,i)}, x_{t-1}^{(\theta,i)}, \theta \right) = \frac{d}{d\theta} \log p \left(y_t \mid x_{t-1}^{(\theta,i)} \right). \tag{4.65}$$

4.7.2.1 Results

In this section, the proposals outlined in Section 4.6 when inferring the parameters of the stochastic volatility model outlined in (3.5) - (3.7) are compared. Each simulation is initialised with $N = 5000$ particles, $T = 500$ observations, $M = 5000$ MCMC iterations (first 2000 discarded as *burn-in*) and uses CRN resampling. Initial values of θ are fixed across all simulations.

The first and second rows of subplots (a), (b) and (c) in Figure 4.3 show the histogram and trace plots of the accepted values of μ, ϕ and σ_v, respectively. These plots provide a visual indication of how well each of the samplers perform. However, they should not be the only measure used to asses convergence. Table 4.3 outlines a number of MCMC diagnostics that determine if the sampler has converged to equilibrium. These include the mean of the posterior samples, which, on its own, is not very informative, especially if the inferred parameter is unknown.

From the trace plots, it is evident that for MALA and HMC, for some values of L, there exists serial correlation between consecutive draws. This results in poor exploration of the parameter space. NUTS has the least serial correlation between MCMC draws. This observation is highlighted in the third row, where auto-correlation function (ACF) plots for the simulations are shown. These plots present the auto-correlation for a Markov chain up to a user-specified number of lags, selected to be 100. The desired ACF plot is large at short lags but quickly drops towards zero. For MALA and a number of the HMC simulations the ACF plots do not reach 0 within the specified 100 lags. The integrated auto-correlation time (IACT) (see Section 3.5.2) is a measure of the area under the ACF plot. This value gives an indication of the mixing within the Markov chain and should therefore be minimised. Table 4.3 shows that using NUTS results in lower IACT scores for parameters μ and σ_v and is comparable with HMC with $L = 6$ for ϕ.

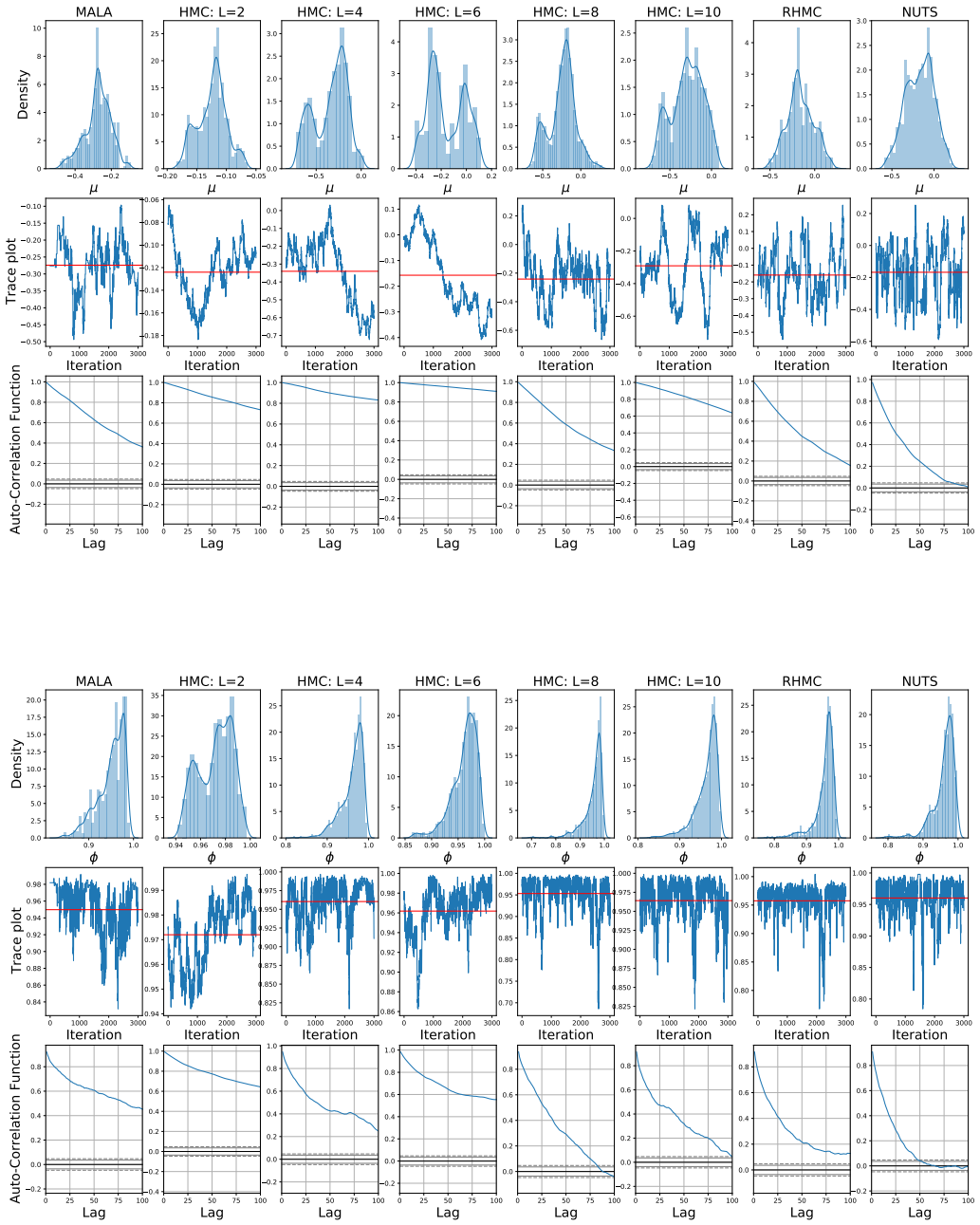


Figure 4.3: Density, trace and auto-correlation function plots for parameters: μ (top) and ϕ (bottom). First row: histograms of posterior estimate. Second row: trace plots and estimated mean (red horizontal line). Third row: ACF plots with lag = 100.

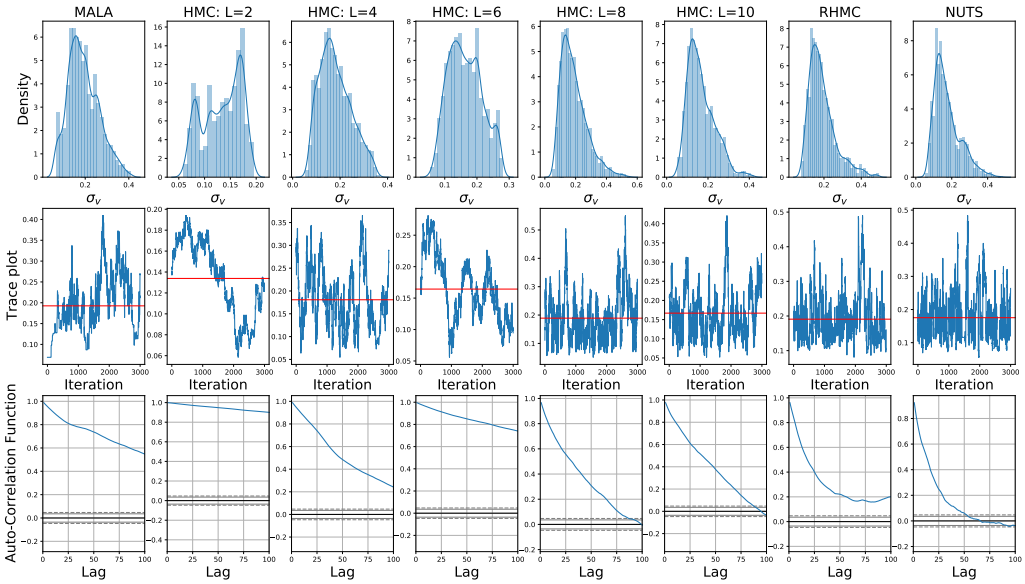


Figure 4.3: Density, trace and auto-correlation function plots for parameters: σ_v . First row: histograms of posterior estimate. Second row: trace plots and estimated mean (red horizontal line). Third row: ACF plots with lag = 100. (Cont)

The ESS is also shown in Table 4.3. Much like the IACT scores for parameters μ and σ_v, NUTS provides better results than the other samplers and is slightly worse than HMC with $L = 6$ for ϕ.

As explained previously, randomising the L parameter within HMC can avoid periodicities in the underlying Hamiltonian dynamics. To do this we draw an L value from an exponential distribution with a mean parameter of 2.5. To ensure this value is an integer we round up. It is evident when looking at Table 4.3, randomising the L parameter in HMC provides better results for μ when compared with HMC with fixed L but is still worse than NUTS.

The mean estimates of θ differ slightly in each simulation to those presented in [68], which were $[-0.23, 0.97, 0.15]$. We believe this disparity stems from [68] using the same particle filter but a MHRW proposal for the parameters. However, when a reparameterised model (described on page 29 of [68]) is used, they obtain estimates equal to $[-0.16, 0.96, 0.17]$, which are very similar to those seen in Table 4.3 when using NUTS, $[-0.17, 0.96, 0.18]$.

		MALA	**HMC**										**RHMC**	**NUTS**
			L1	L2	L3	L4	L5	L6	L7	L8	L9	L10		
Mean	μ	-0.27	-0.12	0.01	-0.34	-0.01	-0.16	-0.20	-0.24	-0.16	-0.30	-0.17	-0.16	-0.17
	ϕ	0.95	0.97	0.94	0.96	0.97	0.96	0.96	0.95	0.95	0.96	0.93	0.96	0.96
	σ_v	0.19	0.13	0.23	0.18	0.14	0.16	0.18	0.19	0.2	0.17	0.23	0.19	0.18
IACT	μ	131	173	190	182	164	192	154	125	163	167	174	101	66
	ϕ	125	158	175	100	98	139	35	68	79	73	144	63	36
	σ_v	148	191	181	111	137	173	44	74	89	81	146	66	35
ESS	μ	15	5	1	2	8	1	12	16	4	11	1	30	50
	ϕ	3	2	2	25	15	8	94	59	37	42	18	75	89
	σ_v	2	1	2	26	6	6	71	50	34	44	16	55	102
	Acc. rate	0.35	0.73	0.81	0.73	0.87	0.83	0.70	0.66	0.76	0.68	0.70	0.53	0.86

Table 4.3: IACT, ESS, mean estimate of θ and acceptance probability of the samplers, applied to the SV model for $N = 1000$ particles and $M = 5000$ MCMC iterations. For each simulation of HMC, RHMC and NUTS the value of ϵ is fixed. CRN resampling is used in all experiments.

4.7.3 Epidemiological Models

The SIR model in Section 3.1.4.1 is extended to include an exposed compartment. The SEIR model follows a similar discrete time approximation to the SIR model. Table 4.4 provides a description of the parameters used in the SIR and SEIR models.

Similar to the stochastic volatility model example described in Section 4.7.2, the proposal distribution is defined to be equivalent to the dynamics, such that

$$q\left(x_t^{(\theta,i)}|x_{t-1}^{(\theta,i)}, y_t\right) = p\left(x_t^{(i)}|x_{t-1}^{(\theta,i)}, \theta\right), \tag{4.66}$$

simplifying the weight update to

$$w_{1:t}^{(\theta,i)} = p\left(y_t|x_t^{(\theta,i)}\right) w_{1:t-1}^{(\theta,i)}. \tag{4.67}$$

The derivative of the log-likelihood is given by (4.4). Due to the simplification in (4.66), the derivative of the particle weights w.r.t θ reduces to

$$\frac{d}{d\theta} w_{1:t}^{(\theta,i)} = \frac{d}{d\theta} p\left(y_t|x_t^{(\theta,i)}\right) w_{1:t-1}^{(\theta,i)} + p\left(y_t|x_t^{(\theta,i)}\right) \frac{d}{d\theta} w_{1:t-1}^{(\theta,i)}. \tag{4.68}$$

As explained in Section 4.2.2, when calculating the derivatives of the particle states it is evident that $\frac{d}{d\theta} p\left(y_t|x_t^{(\theta,i)}\right)$ does not explicitly depend on θ, but $x_t^{(\theta,i)}$ does. Therefore, in order for the derivative of the likelihood w.r.t θ to be non-zero, the derivative $dx_t^{(\theta,i)}/d\theta$ must first be calculated. Defining

$$x_k^{(\theta,i)} = f(x_{t-1}^{(\theta,i)}, \theta, y_t, \epsilon_t^i), \tag{4.69}$$

the derivatives can be calculated using (4.21), (4.22) and (4.23).

As outlined in Chapter 1, MHRW is the commonly chosen proposal within p-MCMC when modelling epidemics. Therefore, for this analysis, the MHRW proposal is compared with NUTS.

4.7.3.1 SEIR Model

The discrete time approximation of the dynamics of the SEIR model is defined as follows:

Parameter	Description	Prior Information
P	The total population	-
s_t	The proportion of people in the susceptible compartment	$1-(e_0+i_0)$
e_t	The proportion of people in the exposed compartment	$\text{Unif}(0.00016, 0.00024)$
i_t	The proportion of people in the infectious compartment	$\text{Unif}(0.00016, 0.00024)$
r_t	The proportion of people in the recovered compartment	-
β	Mean rate of people an infected person infects per day	$\text{HalfNormal}\,(0.0, 0.5)$
γ	The proportion of infected recovering per day	$\text{Normal}(4.0, 5.0)$
δ	Length of incubation period	$\text{Normal}(4.0, 5.0)$
R_0	The total number of people an infected person infects	-

Table 4.4: Table of the parameters and a description used in the statistical SIR and SEIR models. Prior information is also included.

$$s_{t+1} = s_t - \beta i_t s_t + \epsilon_\beta, \tag{4.70}$$

$$e_{t+1} = e_t + \beta i_t s_t - \delta e_t - \epsilon_\beta + \epsilon_\delta, \tag{4.71}$$

$$i_{t+1} = i_t + \delta e_t - \gamma i_t + \epsilon_\gamma - \epsilon_\delta, \tag{4.72}$$

$$r_{t+1} = r_t + -\gamma i_t + \epsilon_\gamma. \tag{4.73}$$

4.7.3.2 Observation Equation

The log of the observations is given by (3.27). For the purpose of this analysis b_l, ϕ_l and σ_l are known which allows the likelihood to be formulated as

$$p\left(y_t | x_t^{(\theta,i)}\right) = \log \mathcal{N}\left(y_t; b_l i_t^{\phi_t}, \sigma_l^2\right). \tag{4.74}$$

4.7.3.3 SIR Results

First, NUTS and MHRW proposals are compared in p-MCMC by inferring the parameters of the SIR model presented in Section 3.1.4.1. The parameter and prior choices are the same as those described in [16]. The dynamics in (7.9) and (7.10) are simulated for $T = 125$ days with a population of $P = 5000$. The true values of $\theta = [\beta, \gamma, v]$ are fixed at 0.254, 0.111 and 1.246, respectively. At t_0, the number of susceptible individuals is 4990 and the number of infected 10. Figure 4.4(a) presents the proportion of the population in

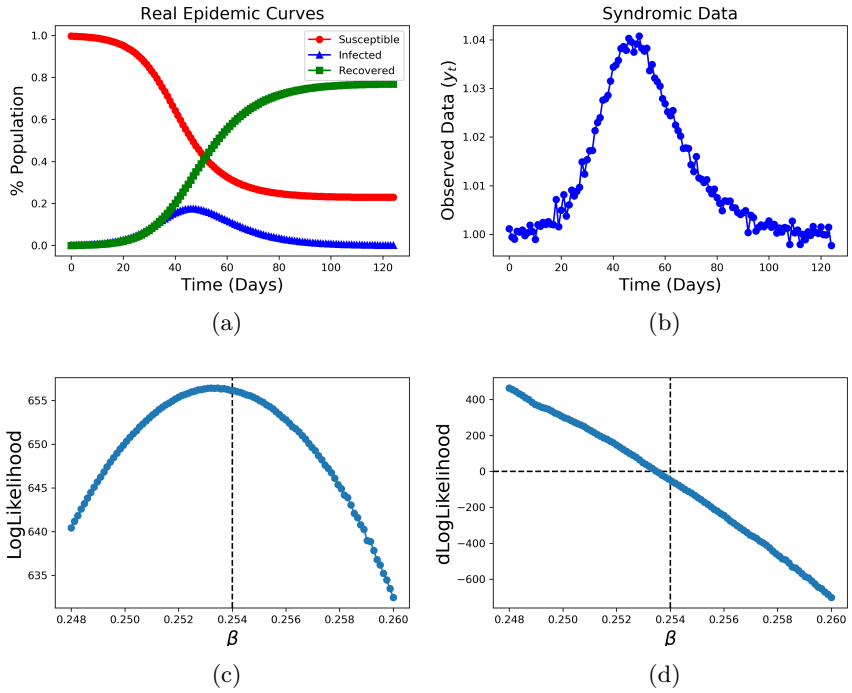


Figure 4.4: Simulated epidemic curves for the SIR model with $\theta = [\beta, \gamma, v]$ fixed at 0.254, 0.111 and 1.246, respectively (a), syndromic data simulated from the infected curve (b) and log-likelihood and gradient of the log-likelihood w.r.t β (c) and (d), respectively. The dashed horizontal line indicates a gradient of 0. True values of β are given by the dashed vertical lines.

each compartment throughout the duration of the epidemic. Figure 4.4(b) presents the simulated syndromic observations, $y_{1:T}$, which are a fraction of the infected compartment. Observations are drawn from (3.27), with parameters b_l, ϕ_l and σ_l fixed at 0.25, 1.07 and 0.0012, respectively.

The particle filter is run to obtain estimates of the log-likelihood and the gradient of the log-likelihood w.r.t β across a range of 100 values equally spaced between 0.248 and 0.260. The log-likelihood should be maximised and a gradient of 0 observed when $\beta = 0.254$. Results are provided in Figure 4.4(c) and (d).

The MSE between true and inferred values of θ for MHRW and NUTS, in addition to the computation time in seconds for different numbers of particles, N, are presented in Table 4.5. The MSE and computation time are averaged over 10 runs. Each run uses a different random number seed and is run for 50 MCMC iterations. Different random

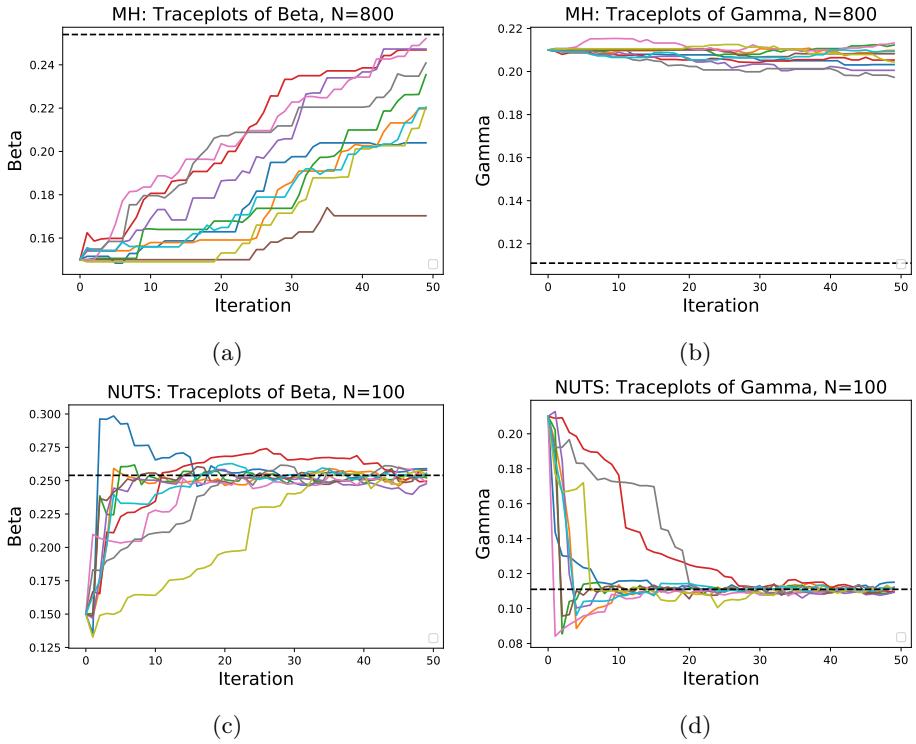


Figure 4.5: Trace plots of β and γ for 10 independent Markov chains with the same initial values but different random number seeds, for MHRW (a)-(b) and NUTS (c)-(d) simulations with similar computation times (see Table 4.5). True values indicated by horizontal dashed line.

number seeds will produce different realisations when resampling occurs (see Section 4.4) and so in turn, will converge to the true value of θ at different rates. Averaging over the 10 runs provides a more accurate representation of the typical convergence accuracy and the computation time. This is exemplified in Figure 4.5 for MHRW (subplots: (a) and (b)) and NUTS (subplots: (c) and (d)). Each trace plot represents an independent MCMC chain with initial values: 0.15 and 0.21 for β and γ, respectively. Note that, for the MHRW proposal, the step sizes of β and γ are as stated in [16]: 0.005 and 0.001, respectively. Table 4.5 shows that NUTS minimises the MSE in less computation time when compared with MHRW.

	β	γ	Time (secs)
N=50			
MHRW	0.0055	0.0095	2.3020
NUTS	**0.0010**	**0.0006**	25.4688
N=100			
MHRW	0.0051	0.0095	3.9031
NUTS	**0.0010**	**0.0006**	31.9671
N=200			
MHRW	0.0054	**0.0094**	7.6374
NUTS	**0.0009**	0.0097	51.1340
N=400			
MHRW	0.0049	0.0097	17.4208
NUTS	**0.0012**	**0.0008**	101.5174
N=800			
MHRW	0.0046	0.0094	39.4373
NUTS	-	-	-

Table 4.5: Comparison of MSE and computation time for MHRW and NUTS proposals for varying numbers of particles, N.

4.7.3.4 SEIR Results

The convergence of the Markov chain to the true parameters when running with different numbers of particles, N, is now presented for the SEIR model in Section 4.7.3.1. Table 4.6 outlines the results when using NUTS with Δ fixed at 0.0055. The simulation is run for 2000 samples, with the first 1000 discarded as *burn-in*. The true values of β, γ and δ are 0.254, 0.111 and 0.400, respectively. Figures 4.6 (a) and (b) are the trace plots for β when $N = 16$ and 256, respectively. It is evident that $N = 256$ results in a Markov chain with better mixing, which in turn explores $\pi(\theta)$ more efficiently than for $N = 16$. This result is supported by the acceptance rates in Table 4.6, where 53% and 20% of samples are accepted for $N = 256$ and $N = 16$, respectively. The corresponding auto-correlation function (ACF) plots are provided in Figures 4.6 (c) and (d). ACF plots present the auto-correlation between samples in the Markov chain as a function of a user-specified lag, selected to be 100. The plot that tends towards 0 in the fewest number of lags is considered superior. This is the case for $N = 256$. Table 4.6 outlines the integrated auto-correlation time (IACT) for each parameter. This is a numerical measure of the area under the ACF

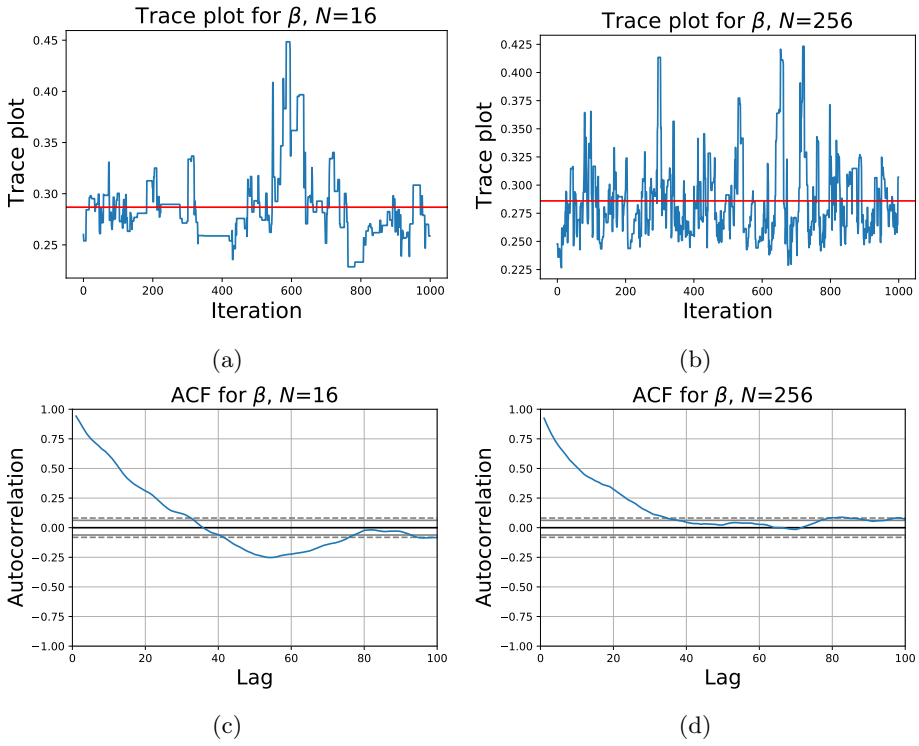
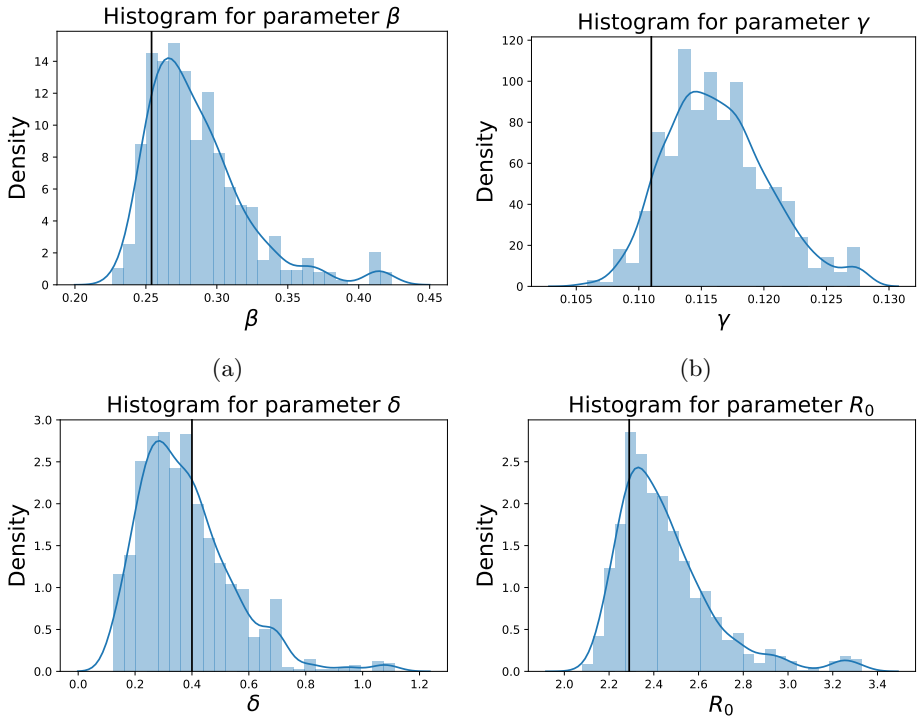


Figure 4.6: Convergence plots for β in the SEIR model when using NUTS, for $N = 16$ and 256: trace plots (a)-(b), ACF plots (c)-(d).

plot, with correlation between consecutive samples decreasing with IACT value. The ESS for varying N and the computation time in seconds are also provided in Table 4.6. Note that the ESS and IACT scores do not monotonically rise and fall, respectively, when N is between 16 and 128. This could be accountable to insufficient increments of N.

The predicted marginal distributions for the parameters β, γ and δ when using $N = 256$ and NUTS are presented in Figures 4.7 (a), (b) and (c), respectively. The predicted distribution around the basic reproduction number, R_t, calculated by $R_t = \beta/\gamma$, is also provided in Figure 4.7 (d).



(a) (b) (c) (d)

Figure 4.7: Marginal distribution plots for β (a), γ (b), δ (c) and R_t (d), when using NUTS and $N = 256$ for the SEIR model.

Num. Particles	16	32	64	128	256
Mean Estimate					
β	0.287	0.273	0.293	0.299	0.286
γ	0.116	0.115	0.117	0.118	0.116
δ	0.368	0.429	0.337	0.321	0.379
IACT					
β	66	16	35	17	2
γ	65	11	35	15	2
δ	69	17	26	9	7
ESS					
β	16	46	21	65	81
γ	15	50	11	74	87
δ	13	38	30	63	73
Time (s)	5952	7798	9985	12831	21763
ESS/s					
β	0.003	0.006	0.002	0.005	0.004
γ	0.003	0.006	0.001	0.006	0.004
δ	0.002	0.005	0.003	0.005	0.003
Acc. Rate	0.20	0.33	0.46	0.52	0.53

Table 4.6: Markov chain convergence statistics for varying N.

4.8 Conclusions and Future Work

This chapter outlines an extension of the *reparameterisation trick* that uses common random numbers when performing the resampling step in a particle filter. Employing this method limits the discontinuities encountered when calculating gradients used in HMC and NUTS. These algorithms were applied to three problems, for which using NUTS was found to improve the mixing of the Markov chain compared to MALA or HMC. Different methods for resampling were compared. It was shown that CRN resampling produces more accurate estimates in shorter run time.

Analysis using Hessian information about the log-posterior within MCMC proposals, as considered in [53], was not included in this chapter. This is due to the generic complexity of computing the second-order partial and full derivatives of the equations in Sections 4.2.2 and 4.3. An estimate of the state-dependent Hessian matrix could be derived, using the gradients estimated in (4.60), via the Gaussian process construction provided in [93]. An avenue for future work is therefore to incorporate the resulting Hessian matrix in the proposal (4.61), in a similar manner to that of [53], or as the mass matrix within NUTS. Doing so could provide additional performance gains over those reported herein.

An additional interesting direction for future work is to perform a broader comparison of algorithms that can be applied to parameter estimation in SSMs. These include, but are not limited to, SMC^2 [94], nudging the particle filter [95] and the nested particle filter [96].

The nested particle filter and SMC^2 have two layers of SMC methods: one (an SMC sampler with N_θ particles) estimates the density function over the static parameters, θ, and the other (a particle filter with N_x particles) considers the dynamic states. The difference between the two methods is that the nested particle filter runs in a purely recursive manner. A detailed comparison of the nested particle filter and SMC^2 can be seen in [96]. The computational complexity of both methods is $O(N_\theta N_x T)$, as in the methods described in this chapter (assuming the simulation is run for N_θ MCMC steps).

In the nudging particle filter, particles are *nudged* towards regions of the state space where the likelihood is deemed to be high. In [95], this method is applied to the p-MCMC algorithm with a MHRW proposal (see Section 3.4). An outline of how gradient nudging steps can be used within the framework of differentiable likelihoods and automatic differentiation libraries is provided. This approach is applicable to the methods proposed in this chapter, with the potential to offer improvements in performance.

Further note that there exists a trade-off between the theoretical concerns related to convergence and the empirical performance achieved. Future work may involve derivation of mathematical proofs of the regularity properties of the estimated derivatives considered in this chapter.

Chapter 5

Particle-NUTS using PyTorch and PyMC3

Chapter 4 outlines that the standard particle filter cannot be differentiated due to the sampling and resampling steps. Recent developments have focused on how to make these inherent operations of the particle filter compatible with automatic differentiation (AD) libraries. In this chapter, a framework for calculating gradients from a particle filter using PyTorch is presented. The resulting gradients can be used within PyMC3's implementation of the No-U-Turn sampler. We compare the proposed algorithm to p-MCMC with the MHRW proposal and show that it samples more effectively and efficiently when inferring the parameters of three SSMs that use real datasets.

5.1 Introduction

Automatic differentiation (AD) efficiently calculates the derivatives of functions with thousands of inputs, making optimisation feasible. Optimisation algorithms, such as stochastic gradient descent, have therefore become a widely used method in the machine learning community. Existing AD frameworks include PyTorch [97], TensorFlow [98], Autograd [99] and Julia [100]. Turing is a ppl that adds a thin layer to Julia code to easily describe models using packages from Julia. A benefit of using Turing over Stan is that critical comparisons of different samplers can be made. For example, Turing can make inferences on continuous and discrete random variables using MHRW, Gibbs, HMC and NUTS. This is in contrast to Stan which can only make inferences on continuous random variables

using NUTS. However, Turing is less mature than Stan so may lack specific tools and functionality that Stan possesses.

PyTorch is primarily a machine learning framework that performs dynamic tensor computations with AD and has been used in applications such as neural networks, computer vision and natural language processing. A computational graph is created that contains the operations used during the executable program in the form of a directed acyclic graph. Two methods for building these graphs include statically and dynamically, which underpin TensorFlow and PyTorch, respectively. Nodes contained within the graph correspond to mathematical operations. According to [101], PyTorch is easier to debug as the error messages are more informed. This is in contrast to Theano where error messages are less helpful. According to [102], PyTorch and Theano are comparative in terms of execution speed.

The particle filter is inherently non-differentiable due to reasons explained in Chapter 4. More specifically, the log-likelihood w.r.t θ, given by (3.41), cannot be differentiated. This is due to the randomness in the sampling step and the discrete operation required for multinomial resampling. The *reparameterisation-trick* outlined in Section 4.2.2 allows the sampling operation to be formulated such that it is compatible with AD.

The novel method of fixing the random number seed within multinomial resampling, which is outlined in Section 4.4, is applicable when using PyTorch. However, as described in Section 4.4.1, calculating the gradient of the log-likelihood w.r.t to the parameters of the particle filter can result in discontinuities. This gives biased gradients [43, 103], which can affect subsequent results in optimisation algorithms, where the aim is for the log-likelihood to be maximised. Sampling algorithms such as p-MCMC can correct for the disparity between the proposal and the target distribution. Differentiable resampling compatible with AD has been an active area of research in machine learning (see Section 4.5). A detailed comparison of these methods when inferring the parameters of SSMs can be seen in Section 4.7. The NUTS sampler used to obtain the results in Section 4.7 is the original algorithm described in [26]. Since its publication in 2014, this algorithm has been extensively developed to include features that have resulted in more effective sampling. A number of these extensions are outlined in Section 5.3.

PyMC3 is a ppl that allows users to define models using Python syntax and to perform Bayesian inference and parameter estimation using NUTS and MHRW. An advantage of using ppls is that the user does not require an in-depth knowledge of the mechanics and processes of MCMC. PyMC3 has the ability to define the log-likelihood and the gradient

of the log-likelihood, needed to sample using NUTS, as *black box* functions [104]. In Section 5.3 a step-by-step process outlining how to calculate the gradient from a particle filter using PyTorch is provided. This process can be passed into PyMC3 NUTS via a custom Theano wrapper.

5.2 Differentiation Methods

Manually calculating the derivatives of non-trivial functions using chain and quotient rules can be extremely time consuming and can result in errors. Three popular methods for calculating the derivative of a function are numeric, symbolic and automatic differentiation. These methods are described in Sections 5.2.1, 5.2.2 and 5.2.3, respectively.

5.2.1 Numerical Differentiation

Numerical differentiation is applied to a function $f(x)$, with parameter x, such that its derivative is defined by $f'(x)$. The simplest method to derive this derivative is to use finite difference approximation, where the gradient of the function $f(x)$ at the point $x = h$ is calculated as follows:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{5.1}$$

When approximating $f'(x)$ in (5.1), the smaller the value of h, the better the approximation. Three methods for estimating derivatives that compare the value of the function at two points (first-order differentiation) are presented in Table 5.1. The truncation error associated with each gradient calculation is also provided. This error is the difference between the analytical and numerical gradients. Numerical differentiation can be inefficient for high dimensional functions due to the evaluation of n-dimensional gradients.

5.2.2 Symbolic Differentiation

Symbolic differentiation calculates the derivative of a function w.r.t its parameters and outputs its mathematical expression. Unlike when calculating derivatives numerically, where the output is an evaluation of the derivative at a given point, the output of symbolic differentiation can be evaluated across the domain of the function. Symbolically, $\frac{df(x,y)}{dx}$

Method	Formula	Error
Forward	$f'(x) = \frac{f(x+h)-f(x)}{h}$	$\mathcal{O}(h)$
Backward	$f'(x) = \frac{f(x)-f(x-h)}{h}$	$\mathcal{O}(h)$
Central	$f'(x) = \frac{f(x+h)-f(x-h)}{2h}$	$\mathcal{O}(h^2)$

Table 5.1: Methods for numerical differentiation.

can be expressed and evaluated at $x = 1$, $y = 0.5$ and then at $x = 1.3$, $y = 0.9$ with no recalculation. This differs from AD (see Section 5.2.3). However, non-trivial functions can result in overly complicated symbolic representations of the derivative which can be expensive to evaluate.

5.2.3 Automatic Differentiation

AD is a set of techniques that enables automatic computation of derivatives of continuous functions. Computation of the derivative is undertaken in parts and the flow of the operations stored in a computational graph. Inputs and outputs are represented as nodes that are connected by the operations joining the nodes. One benefit of storing the flow of operations in a computational graph is that control flow statements, such as for loops and if statements, can be used within the program. This differs from symbolic differentiation (see Section 5.2.2).

AD can be performed in forward or reverse mode (often referred to as backpropagation [105]). As operations are performed and the computational graph is created, forward mode AD simultaneously calculates the derivatives and stores them in a derivative trace. This is in contrast to reverse mode, where the derivative is calculated in two stages. Firstly, during the *forward pass*, all variables are evaluated using the relevant mathematical operators and are stored in memory. Secondly, the *backward pass* uses the chain rule to evaluate the derivatives at each node, working backwards from the output. Two considerations when choosing between forward and reverse mode are the memory storage required and the computation time taken to evaluate the derivatives.

5.3 Particle-MCMC and PyMC3

In the subsequent sections, when discussing the particle filter, MHRW, NUTS, p-MCMC with a MHRW proposal and p-MCMC with a NUTS proposal, we refer to the descriptions in Sections 3.2, 3.3.1, 3.3.3, 3.4 and Chapter 4, respectively.

The advancements of NUTS in PyMC3 compared to the original implementation in 2014 include multinomial sampling in place of slice sampling, where each leaf node in the tree is attributed to a transition probability [106]. One benefit of this change is that sampling typically occurs deeper into the tree. The analysis involving NUTS in Chapter 4 did not use step size adaption and specified the identity matrix as the mass matrix within NUTS. However, in PyMC3 the step size is optimised during three stages of *burn-in*: stages 1 and 3 involve a fast adaption period, while stage 2 is slow. Both the fast and slow periods involve adapting the step size, however, the slow period is also used to learn the co-variance associated with the mass matrix. A thorough explanation is provided in [107]. PyMC3 also allows for the step size to be "jittered" during sampling. This benefits the sampling procedure when manoeuvring around regions of high curvature.

An additional advantage of using PyMC3 is that it facilitates definition of the log-likelihood and the gradient of the log-likelihood as external functions. A simple graphical representation is presented in Figure 5.1. Probability distributions or model functions that are not provided by PyMC3 or written in different programming languages, for example C or PyTorch, can therefore be used. An example of coding the particle filter described in Section 3.2 in PyTorch is provided in Appendix C.1. The marginal likelihood of the parameter set θ is calculated by summing the particle filter weights in the approximation in (3.41). By declaring that θ requires a gradient, `requires_grad=True`, and running the `.backward()` method on the marginal log-likelihood, the gradients of θ are calculated. An example of this process can be seen in Appendix C.2.

The workflow is simplified into four distinct processes as follows:

(i) Define the particle filter in PyTorch with inputs θ and $y_{1:T}$ and output $\log p(y_{1:T}|\theta)$ (see Appendix C.1).

(ii) Define a wrapper function that includes the particle filter in (i) and runs `.backward()` on $\log p(y_{1:T}|\theta)$ to obtain $\nabla \log p(y_{1:T}|\theta)$ (see Appendix C.2).

(iii) Define a *black box* function that provides θ to the wrapper function in (ii) and outputs $\log p(y_{1:T}|\theta)$ and $\nabla \log p(y_{1:T}|\theta)$ to PyMC3 NUTS (see Appendix C.3).

(iv) Define the PyMC3 model that calls the *black box* function every time a gradient evaluation is made within NUTS (see Appendix C.4).

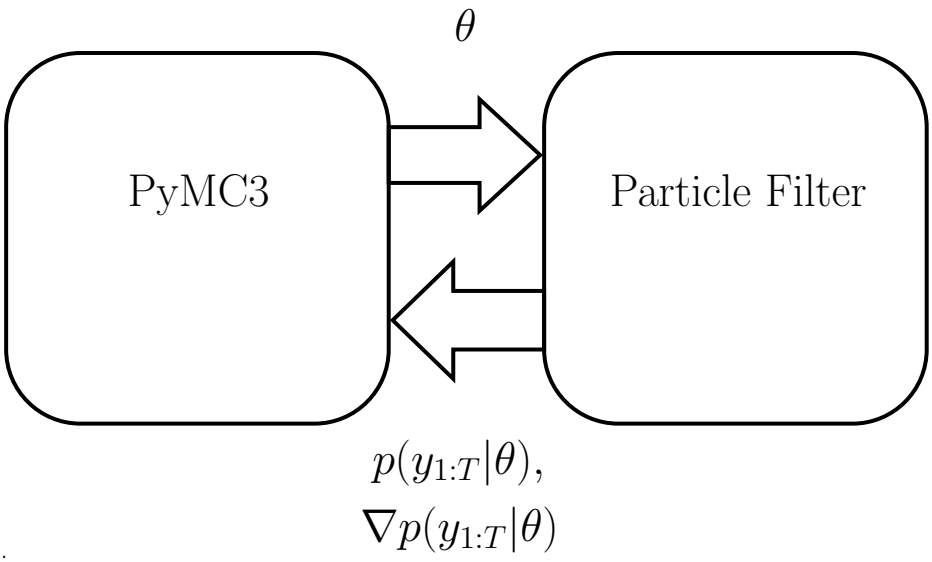


Figure 5.1: A graphical representation of PyMC3's external function. PyMC3 provides θ to the particle filter (Algorithm 6) which in turn calculates $p(y_{1:T}|\theta)$ and $\nabla p(y_{1:T}|\theta)$.

5.4 Examples and Results

In the experiments below, the particle filter is set-up with 128 particles and the dynamic model is used as the proposal (see Section 3.36). MCMC is run for 2000 samples, where the first 1000 are discarded as *burn-in*. NUTS has a target accept probability of 0.8.

5.4.1 Stochastic Volatility and Earthquake Count Models

The model, parameters and priors are those presented in Sections 3.1.2 and 3.1.3, for the SVM and earthquake count models, respectively.

5.4.1.1 Results

Table 5.2 presents a quantitative comparison of the results obtained when using MHRW and NUTS proposals within the PyMC3 sampler. The mean estimates for both samplers are similar, however, using NUTS provides better mixing and is more efficient in terms of effective samples per second.

For the stochastic volatility model, the IACT scores are smaller when using NUTS, with the largest disparity between the two observed for the parameter σ_v, with IACT scores of 65 and 2 for MHRW and NUTS, respectively. This trend continues when considering the results of the earthquake model. The IACT scores for ϕ and μ are over 4 times smaller when using NUTS. For σ, the result is 3 times smaller. This is exemplified in Figure 5.2. The ACF plot shows the auto-correlation for a Markov chain up to a user-specified number of lags, selected as 100 in this case. A more efficient Markov chain will drop towards 0 after fewer lags. This can be seen for all parameters when using NUTS.

NUTS is also more efficient at drawing samples compared with MHRW in the stochastic volatility model and has more than 3 times the efficiency of MHRW when drawing accepted samples in the earthquake model. This improvement is to be expected as using gradient information about the log-posterior will reduce the correlation between successive sampled states by proposing moves to distant states in $\pi(\theta)$ and maintain a high probability of acceptance.

5.4.2 SIR Disease Model

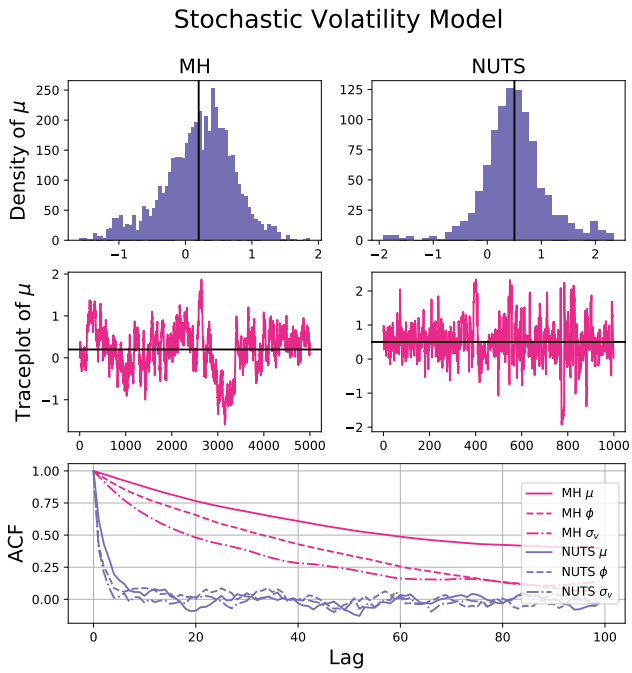
The stochastic SIR model is that presented in Section 3.1.4. In the experiments below a Gamma(2,2) prior is placed on both β and γ.

5.4.2.1 Results

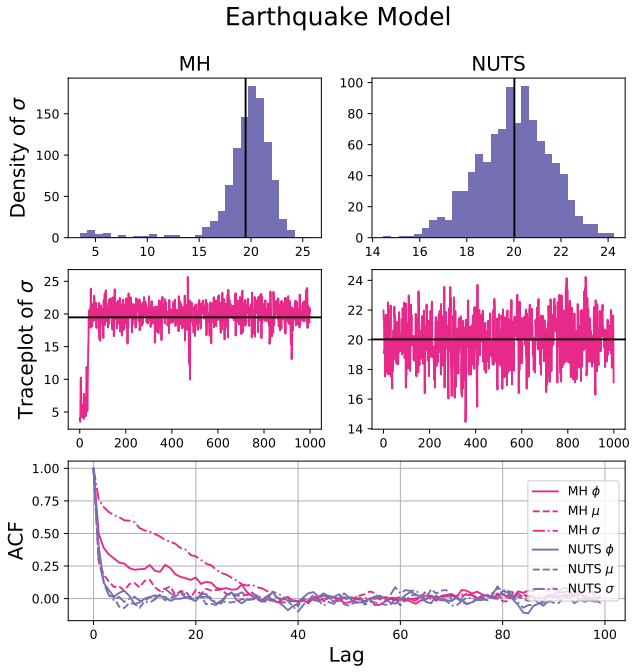
Firstly, data is simulated with known parameters to observe whether they are recovered by the proposed algorithm. The parameters β and γ are fixed at 1.8 and 0.5, respectively, which results in an R_t number of 3.6 (see Table 5.3). The simulation is run for 20 days and the total population is 763, with 1 initial infected person. The resulting epidemic curves are presented in Figure 5.3 (a). As the population is relatively small, we opt to use a Poisson likelihood [108]. The incidence data represents a proportion of the infected compartment and is simulated via Poisson(I_t). The corresponding curve can be seen in the top right of Figure 5.3 (a). The marginal plots of β and γ are presented in Figure 5.3 (a). Table 5.3 shows that the estimated marginals encapsulate the true values. Running 4 independent chains passed the threshold for $\hat{R}$ with no divergent transitions when running NUTS. This is exemplified in Figure 5.3 (a), where the mean of $\log(\beta)$ and $\log(\gamma)$ are plotted against their true values. Divergence has been shown to deviate from the corresponding true value [109].

Stochastic Volatility Model				
	Mean Estimate	IACT	ESS	ESS/s
MHRW				
μ	0.20±0.53	120	23	0.09
ϕ	0.93±0.04	81	59	**0.23**
σ_v	0.36±0.40	65	57	0.22
NUTS				
μ	0.50±0.60	**5**	**225**	**0.21**
ϕ	0.94±0.04	**6**	**146**	0.14
σ_v	0.20±0.11	**2**	**375**	**0.35**
Earthquake Model				
	Mean Estimate	IACT	ESS	ESS/s
MHRW				
ϕ	0.56 ±0.16	17	104	0.04
σ	0.24 ±0.04	9	174	0.07
μ	19.48±3.09	29	106	0.04
NUTS				
ϕ	0.53 ±0.13	**4**	**423**	**0.23**
σ	0.24 ±0.04	**3**	**504**	**0.27**
μ	20.02±1.53	**5**	**502**	**0.27**

Table 5.2: Results for the stochastic volatility model of Section 3.1.2 and the earthquake model of Section 3.1.3 when using MHRW and NUTS as the proposal.



(a)



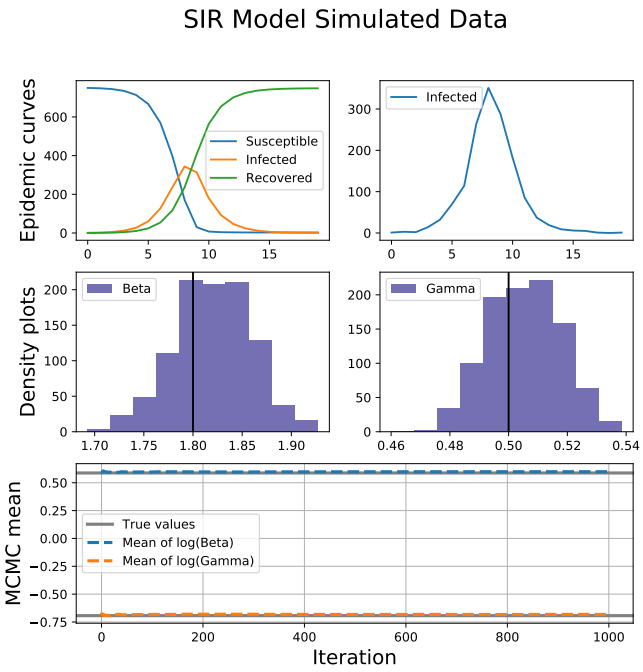
(b)

Figure 5.2: Comparison of results for (a) stochastic volatility model of Section 3.1.2 and (b) earthquake model of Section 3.1.3 when using MHRW and NUTS as the proposal.

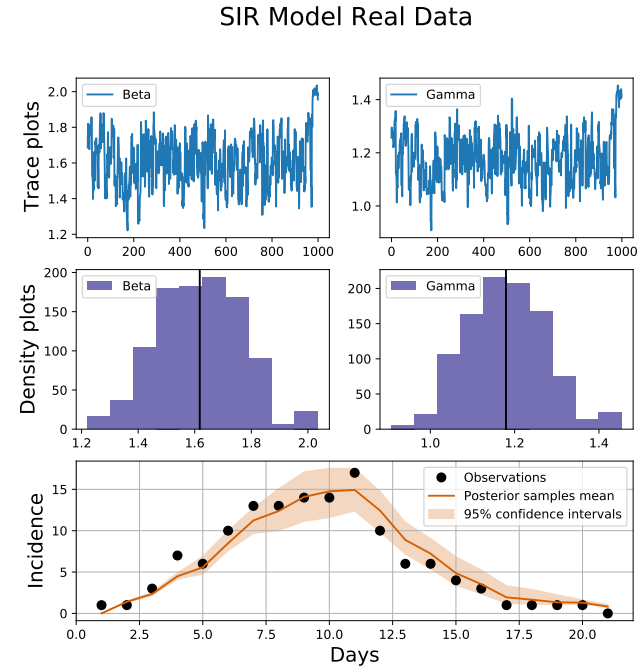
Simulated Data			
	β	γ	R_t
True Value	1.80	0.50	3.60
Estimated	1.82±0.04	0.51±0.01	3.61±0.12
Real Data			
	β	γ	R_t
Estimated	1.62±0.14	1.18±0.09	1.37±0.05

Table 5.3: Results for the SIR model in Section 3.1.4 when using NUTS as the proposal.

The same inference is undertaken with a real dataset consisting of common cold infections on Tristan da Cunha [108, 110]. This is a useful example, as the population on the island is closed. The total population is 300, with 1 infected person at t_0. The marginal plots of β and γ in addition to the model fit plots for inferred incidence from the particle filter are provided in Figure 5.3 (b). Table 5.3 outlines the estimated values of β and γ as 1.62±0.14 and 1.18±0.09, respectively. These results are within the range of those inferred in [108].



(a)



(b)

Figure 5.3: Results for the SIR model in Section 5.4.2 when simulating data in (a) and using real data in (b), with NUTS as the proposal.

5.5 Conclusions and Future Work

In this chapter, a framework for computing $\nabla \log p(y_{1:T}|\theta)$ from a particle filter that can be used by PyMC3 and NUTS is provided. Comparisons between MHRW and NUTS proposals were made when inferring the parameters of three SSMs using real data.

In each of the three examples, the number of observations is small. This is because the computational graph becomes large for large datasets, making gradient calculation time consuming. As NUTS makes 2^j leapfrog steps per MCMC iteration, the combined computational time required for calculating gradients can become large, given the high computation time for a single gradient evaluation. As an example, the computation time required for evaluation of the gradient in Section 3.1.2 is presented in Figure 5.4. Computation time increases with increasing observations. If NUTS is run with a tree depth of 8, 256 gradient evaluations are needed. This can result in a single MCMC iteration taking 4.7 minutes to complete, decreasing the efficiency of the sampler.

A direction for future work is to run some aspects of the particle filter on a GPU. PyTorch has in-built functionality that allows users to execute computations in parallel. Future work will also involve incorporating the differentiable particle filter within SMC^2 [94]. The method for differentiating the particle filter presented in this chapter and in Chapter 4 could be incorporated into the SMC sampler layer, which has NUTS as the proposal, as seen in [111].

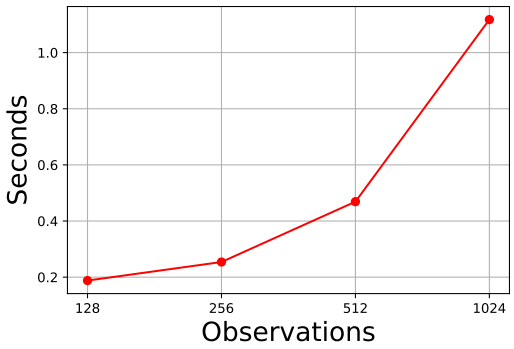


Figure 5.4: Computation time required for increasing observations.

Chapter 6

Refining Epidemiological Forecasts with Simple Scoring Rules

Infectious disease model estimates constitute a significant part of the scientific evidence used to inform the response to the COVID-19 pandemic in the UK. These estimates vary strikingly in their bias and variability. Epidemiological forecasts should be consistent with future observations. In this chapter, simple scoring rules are used to refine the forecasts of a novel statistical model for multisource COVID-19 surveillance data by tuning its smoothness hyperparameter. The usefulness of the normalised estimation error squared (NEES) scoring rule in determining whether estimates are over-confident or over-cautious is also highlighted.

6.1 Introduction

Forecasts relating to COVID-19 are essential for informing short- and long-term public health decisions. However, forecasts can vary significantly depending on the statistical model and the data used for calibration [65, 112, 113, 114, 115, 116]. Examples when estimating the reproductive number R_t in the UK are presented in Figure 1 of both [64] and [117]. Models using different datasets produce estimates with varying degrees of uncertainty. It can therefore be challenging for public health officials to distinguish the forecast that best represents reality. This can complicate decisions on measures such as school closures, limiting individual contacts and implementing lockdowns.

Section 3.1.4 outlines the assumptions made when utilising the SIR model for disease

modelling. Extensions to this simple compartmental model include age stratification [72, 118], differentiating between age group contacts [72, 118], household transmission models [118], simultaneous simulation of the spread of disease in different regions [72, 118, 119] and the inclusion of interventions such as vaccination status [118] and non-pharmaceutical data [119]. The SIR model can also be extended to model the capacity of healthcare setting, as seen in [120].

In the UK, a joint effort has been undertaken to produce estimates of R_t, with notable examples presented in [64, 65]. Data used for this purpose includes death and non-pharmaceutical intervention data [119], laboratory-confirmed COVID-19 diagnoses [72], the UK's National Health Service (NHS) Pathways data [121], hospital admissions data [118] and positive test results [122]. An example of the differences in estimating R_t using the models in [72, 120] is presented in [123]. Estimates of the R_t number are beneficial in understanding the spread of COVID-19. However, given that R_t is unobservable, the accuracy of these estimates cannot be evaluated. Producing estimates of observable variables allows for the use of scoring rules [124] to critically evaluate forecasts. Providing these metrics with forecasts gives credence to the effectiveness of a statistical model. Examples of observable variables that have been forecast include predicted deaths, hospital admissions and intensive care unit (ICU) occupancy [125]; deaths, hospital admissions and ICU occupancy [118]; daily hospital admissions [120] and daily case counts [122].

Quantile forecasts are one method for producing forecasts. Future observed variables are random and unknown and can be represented by a forecast distribution. Quantile forecasts are made at various quantiles of the forecast distribution. For example, if the forecast distribution is assumed to be normally distributed, the 90% prediction interval is defined by the 5% and 95% quantiles of the forecast distribution. A collection of quantile forecasts from an ensemble of statistical models in the US and European nations are presented in [113] and [126], respectively. Both articles use the weighted interval score (WIS), which is an approximation of the continuous ranked probability score (CRPS) that provides a measure of the absolute distance between forecast and observation. Reference [113] outlines that variability when predicting COVID-19 related deaths between different models is high. Use of the WIS of quantile forecasts of deaths, hospital admissions and total hospital beds occupied is presented in [125]. It is argued in [127] that performing a transformation of forecasts to the log scale before applying WIS and CRPS can mitigate issues that arise when comparing forecasts that contain different orders of magnitude. This can affect scoring rules that are based on absolute error.

Forecasts can also be produced using probabilistic forecasts, where the entire distribution of forecasted values is used. Probabilistic forecasts can be produced using Bayesian methods. One such method involves describing the statistical model in the ppl Stan. Stan uses the high performance NUTS sampler to infer the parameters of the model by sampling and calibrating to disease specific data. The ensemble of forecasts from the samples therefore makes up the probabilistic forecasts. Examples when using Stan for disease modelling include estimation of R_t for influenza [128], R_t for COVID-19 [119], infection severity rates for COVID-19 [129] and COVID-19 deaths [130]. One benefit of Stan is that it generates a summary of the sampling statistics described in Section 3.5. This allows the user to know whether the samples are statistically sound and are sampled correctly.

The simple scoring rule NEES is a popular method in the field of signal processing and tracking for assessing uncertainties associated with forecasts [131]. NEES has been used to argue the merits of extensions of the Kalman filter to specific non-linear settings, where the extended Kalman filter is routinely overconfident [132]. The metric determines whether the estimated variance of forecasts differs from the true variance. If the estimated variance is larger than the true variance, the forecast is over-cautious and if the estimated variance is smaller than the true variance, it is over-confident. This reasoning gives NEES an advantage over other scoring rules. However, NEES is seldom used when forecasting in the context of epidemiology. In this chapter, scoring rules are used to evaluate forecasts made by an epidemiological model that is parameterised differently. Evaluating forecasts using these scoring rules can help to diagnose model limitations.

6.2 Statistical Model

In the analysis of this chapter, the extended SIR disease transition model outlined in the subsequent subsections is implemented in the ppl Stan. NUTS is used to infer the parameters, nowcast the number of people in each compartment and to calibrate to disease specific data. The statistical model can be described in two succinct parts: the transmission model outlines how individuals migrate from one compartment to the next and the observation model states how surveillance data links to the transmission model in the form of calibration.

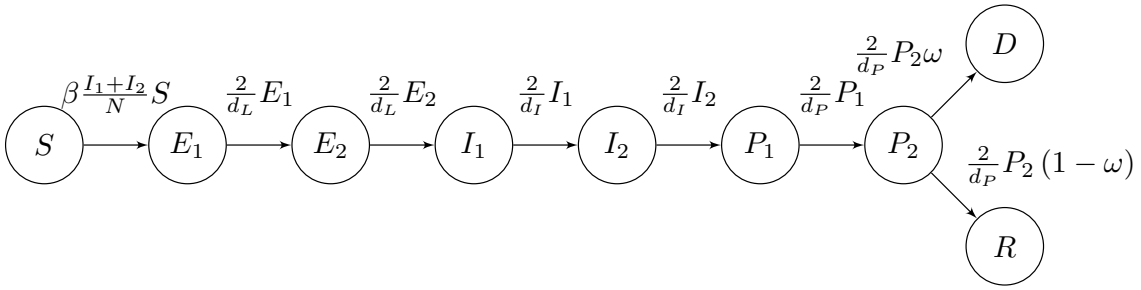


Figure 6.1: Flow diagram of the transmission model illustrating the movement of individuals between states.

6.2.1 Transmission Model

The transmission model in Figure 6.1 is an extension of the simple SIR model outlined in Section 3.1.4. This extension additionally captures two exposed (E), infected (I) and pending (P) compartments, one dead (D) and one recovered (R) compartment. The splitting of the E, I, P compartments into two substates is inspired by [72]. Under this specification, the amount of time spent in each state follows an Erlang rather than an Exponential distribution. The assumptions made in this model are those discussed in Section 3.1.4, i.e. homogeneous mixing of individuals, a closed population and only one region/location is considered. Justification of these assumptions is outlined in Section 6.6. At time t_0, each individual within the population N is assigned to a compartment via the set of initial

conditions:

$$S(0) = (N-5)\,\alpha_1 + 1, \tag{6.1}$$

$$E_1(0) = \frac{1}{2}(N-5)(1-\alpha_1)\,\alpha_2 + 1, \tag{6.2}$$

$$E_2(0) = \frac{1}{2}(N-5)(1-\alpha_1)\,\alpha_2 + 1, \tag{6.3}$$

$$I_1(0) = \frac{1}{2}(N-5)(1-\alpha_1)(1-\alpha_2) + 1, \tag{6.4}$$

$$I_2(0) = \frac{1}{2}(N-5)(1-\alpha_1)(1-\alpha_2) + 1, \tag{6.5}$$

$$P_1(0) = 0, \tag{6.6}$$

$$P_2(0) = 0, \tag{6.7}$$

$$R(0) = 0, \tag{6.8}$$

$$D(0) = 0. \tag{6.9}$$

At least one individual is assigned to the susceptible, exposed and infected compartments, with no individuals attributed to pending, recovered or dead. The number of individuals initially susceptible and infected are determined by the parameters α_1 and α_2, respectively. Over time, infectious individuals make contact with those in the susceptible compartment via mixing within the population. This results in a proportion of the susceptible becoming exposed to the virus. After the latent period of infection, governed by d_L, has elapsed, an individual becomes infectious and migrates to the pending compartment. Infected individuals then go on to recover or die from the virus.

The set of ODEs that govern this process for $t > 0$ is presented below:

$$\frac{dS(t)}{dt} = -\beta(t)\frac{I_1(t) + I_2(t)}{N}S(t), \tag{6.10}$$

$$\frac{dE_1(t)}{dt} = \beta(t)\frac{I_1(t) + I_2(t)}{N}S(t) - \frac{2}{d_L}E_1(t), \tag{6.11}$$

$$\frac{dE_2(t)}{dt} = \frac{2}{d_L}\big[E_1(t) - E_2(t)\big], \tag{6.12}$$

$$\frac{dI_1(t)}{dt} = \frac{2}{d_L}E_2(t) - \frac{2}{d_I}I_1(t), \tag{6.13}$$

$$\frac{dI_2(t)}{dt} = \frac{2}{d_I}\big[I_1(t) - I_2(t)\big], \tag{6.14}$$

$$\frac{dP_1(t)}{dt} = \frac{2}{d_I}I_2(t) - \frac{2}{d_P}P_1(t), \tag{6.15}$$

$$\frac{dP_2(t)}{dt} = \frac{2}{d_P}\big[P_1(t) - P_2(t)\big], \tag{6.16}$$

$$\frac{dR(t)}{dt} = \frac{2}{d_P}P_2(t)\big[1 - \omega\big], \tag{6.17}$$

$$\frac{dD(t)}{dt} = \frac{2}{d_P}P_2(t)\omega. \tag{6.18}$$

Table 6.1 gives a description of the parameters of the ODEs and their priors. The effective contact rate parameter, $\beta(t)$, describes the mean rate of infected individuals per unit time that a susceptible individual comes in to contact with. This is estimated by

$$\beta(t) = \sum_{j=1}^{J}\beta_j(t)\chi_{[t_{j-1},t_j)}(t), \tag{6.19}$$

where $\beta_j(t)$ is the mean rate of contacts per unit time in the jth time interval, defined as follows:

$$\beta_j(t) = \frac{\beta_{j+1} - \beta_j}{t_j - t_{j-1}}(t - t_{j-1}) + \beta_j, \tag{6.20}$$

and

$$\chi_{[t_{j-1},t_j)}(t) = \begin{cases} 1 & \text{if } t \in [t_{j-1}, t_j), \\ 0 & \text{if } t \notin [t_{j-1}, t_j). \end{cases} \tag{6.21}$$

The time between the first two mean contact rates, $\beta_2 - \beta_1$ in (6.20), is the time between the beginning of the observation and the date of the first lockdown. Subsequent time

Parameter	Description	Prior
S	Susceptible compartment	-
E	Exposed compartment	-
I	Infected compartment	-
P	Pending compartment	-
R	Recovered compartment	-
D	Dead compartment	-
N	Total population	-
d_L	Mean time from infected to infectious	$\text{Normal}^+(4.0, 3.0)$
d_I	Mean time infectious	$\text{Normal}^+(5.0, 4.0)$
d_P	Mean time in pending compartment	$\text{Normal}^+(13.0, 4.0)$
α_1	Proportion initially susceptible	$\text{Beta}\,(5.0, 0.5)$
α_2	Proportion initially infected	$\text{Beta}\,(1.1, 1.1)$
ω	Infection fatality ratio (IFR)	$\text{Beta}\,(5.7, 624.1)$
β_1	Initial mean rate of contacts between individuals	$\text{HalfNormal}\,(0.0, 0.5)$
$\beta_2, ..., \beta_J$	Mean rate of contacts between individuals	$\text{Normal}^+\,(\beta_{i-1}, \sigma_\beta)$
$\frac{1}{\phi_{\text{deaths}}}, \frac{1}{\phi_{\text{admissions}}}, \frac{1}{\phi_{\text{calls}}}$	Negative binomial parameters	$\text{Exponential}\,(5.0)$
$\rho_{\text{admissions},k}$, $\rho_{\text{calls},l}$	Ratio of hospital admissions to potential patients	$\text{Beta}\,(1.1, 1.1)$

Table 6.1: Description of the model parameters and their prior distributions. The symbol $^+$ indicates a distribution with its lower tail truncated at zero.

periods, $\beta_{j+1} - \beta_j$ for $j > 1$, are set to seven days. The effective contact rate $\beta(t)$ is a continuous piecewise linear function of time.

6.2.2 Observation Model

As explained in Section 6.1 different research groups use different datasets to calibrate their statistical models. The observation model describes the link between the unobservable transmission model states and the observable data. Serological and death data are assumed to follow binomial and negative binomial distributions, respectively, in [72]. Hospital admissions, ICU occupancy and death counts are either Poisson distributed or binomially distributed for quantities that have a relatively low upper bound in [118]. The analysis undertaken in this chapter assumes that death, hospital admission and 111 call data follow a negative binomial distribution with mean parameter derived from the transmission model and a data specific overdispersion parameter. Examples on how to derive the death and hospital admissions negative binomial distributions can be found in Sections 6.2.2.1 and 6.2.2.2, respectively.

In the subsequent sections, the observation distributions for the surveillance datasets used in this analysis are outlined. Due to the proposed model's flexibility, this set-up is

also applicable to other geographic location and surveillance datasets. An example of this is presented in Chapter 7.

6.2.2.1 Death Data

Death data used in this analysis was downloaded from the UK government website [133]. Aggregated death counts contain the individuals that died within 28 days of receiving a positive COVID-19 test result. This data can be seen in Figure 6.2 (a). The number of COVID-19 related deaths, $d_{\text{obs}}(t)$, observed on day t is assumed to follow the negative binomial distribution[1]

$$d_{\text{obs}}(t) \sim \mathcal{NB}(d(t), \phi_{\text{deaths}}), \tag{6.22}$$

parameterised by an overdispersion parameter ϕ_{deaths}. The mean, $d(t)$, is the difference between the count of individuals within the unobservable D compartment between times $t-1$ and t, such that $d(t) = D(t) - D(t-1)$. The observation begins on 24th March 2020.

6.2.2.2 Hospital Admission Data

Hospital admission data used in this analysis was downloaded from the UK government website [134]. Aggregated admission counts contain the daily COVID-19 related hospital admissions and the total number of COVID-19 patients. This data can be seen in Figure 6.2 (b). The number of COVID-19 related hospital admissions, $h_{\text{obs}}(t)$, on day t is assumed to follow the negative binomial distribution

$$h_{\text{obs}}(t) \sim \mathcal{NB}(h(t), \phi_{\text{admissions}}) \tag{6.23}$$

with

$$h(t) = \rho_{\text{admissions}}(t) \times \frac{2}{d_I} I_2, \tag{6.24}$$

where $\rho_{\text{admissions}}(t)$ is the ratio of hospital admissions to potential patients and $2I_2/d_I$ is the number of new members of the pending state. Similarly to $\beta(t)$, $\rho_{\text{admissions}}(t)$ is a continuous piecewise linear function, where

[1]Note that, the alternative parameterisation of the negative binomial distribution in [73] is used.

$$\rho_{\text{admissions}}(t) = \sum_{k=1}^{K} \rho_{\text{admissions},k}(t)\chi_{[t_{k-1},t_k)}(t). \tag{6.25}$$

In the kth time interval, the ratio of hospital admissions to potential patients, $\rho_{\text{admissions},k}(t)$, is given by

$$\rho_{\text{admissions},k}(t) = \frac{\rho_{\text{admissions},k+1} - \rho_{\text{admissions},k}}{t_k - t_{k-1}}(t - t_{k-1}) + \rho_{\text{admissions},k}. \tag{6.26}$$

The function $\chi_{[t_{k-1},t_k)}(t)$ is defined analogously to the indicator function in (6.19), where for the kth time interval,

$$\chi_{[t_{k-1},t_k)}(t) = \begin{cases} 1 & \text{if } t \in [t_{k-1}, t_k), \\ 0 & \text{if } t \notin [t_{k-1}, t_k). \end{cases} \tag{6.27}$$

All time intervals for hospital admissions are set to twelve weeks. The observation begins on 24th March 2020.

6.2.2.3 111 Call Data

111 call surveillance data used in this analysis was downloaded from the UK government website [135]. Aggregated call counts contain the individuals that reported potential COVID-19 symptoms through NHS Pathways telephone service. This data can be seen in Figure 6.2 (c).

The number of daily 111 calls, $c_{\text{obs}}(t)$, on day t is assumed to follow the negative binomial distribution

$$c_{\text{obs}}(t) \sim \mathcal{NB}(c(t), \phi_{\text{calls}}), \tag{6.28}$$

where $c(t)$ is the mean number of assessment calls on day t. This parameter is given by the product of the ratio of symptom reporters to potential 111 calls, $\rho_{\text{calls}}(t)$, and the sum of the number of new individuals in the infectious and pending states, such that

$$c(t) = \rho_{\text{calls}}(t) \times \left(\frac{2}{d_L}E_2 + \frac{2}{d_I}I_2\right). \tag{6.29}$$

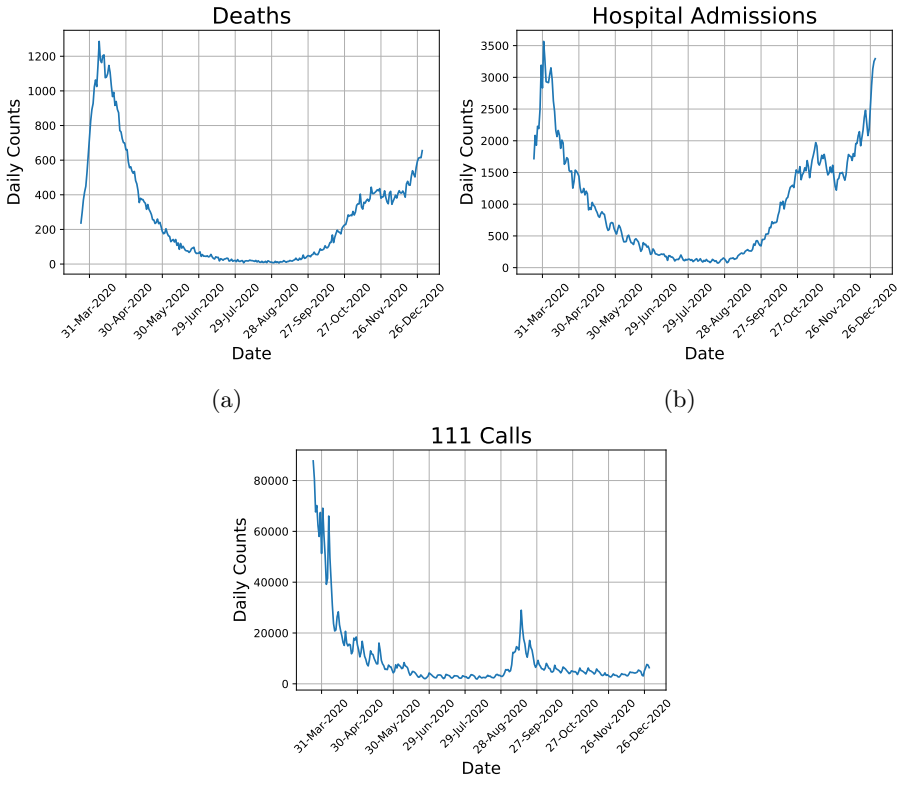


Figure 6.2: Time series of (a) deaths, (b) hospital admissions and (c) 111 calls in England from 24th March 2020 to 31st December 2020.

The ratio $\rho_{\text{calls}}(t)$ is defined as for hospital admissions. The associated definitions (6.25), (6.26) and (6.27) corresponding to call data are therefore not presented here. For 111 calls, all time intervals are set to four weeks. The observation begins on 24th March 2020. A strong correlation between numbers of 111 symptom reports and COVID-19 death reported four weeks later has been observed [121].

6.3 Scoring Rules

The simple scoring rules described in Table 6.2 are methods for evaluating a probabilistic forecast produced by a statistical model. The observed data is compared against the forecast, resulting in a numerical score that is used to determine whether the model represents

Scoring Rule	**Definition**	**Reference**
Logarithmic	$\text{logs}(P, x) = -\log p_x$	Good [137]
Quadratic	$\text{qs}(P, x) = -2p_x + \|p\|^2$	Wecker [138]
Spherical	$\text{sphs}(P, x) = -\frac{p_x}{\|p\|}$	Czado, Gneiting and Held [124]
Ranked probability	$\text{rps}(P, x) = \sum_{k=0}^{\infty} \{P_k - \mathbf{1}(x \leq k)\}^2$	Epstein [139]
Squared error	$\text{ses}(P, x) = (x - \mu_P)^2$	Czado, Gneiting and Held [124]

Table 6.2: Scoring rules utilised in this chapter, where p_x is the probability mass of the predictive distribution for an observed count x, the operator $\|p\|^2 = \sum_{k=0}^{\infty} p_k^2$, P_k is the value of cumulative predictive distribution for a count k, $\mathbf{1}(\cdot)$ is the indicator function, and μ_P and σ_P^2 are the mean and variance of the predictive distribution.

reality. It is argued in [136] that a scoring rule is proper if the expected score for an observation drawn from the forecast is maximised. In addition, scoring rules are strictly proper if the maximum is unique.

6.3.1 Normalised Estimation Error Squared

The NEES score is defined by

$$\text{NEES} = \frac{1}{N} \sum_{i=1}^{N} (x^i - y^i)^T C^{i-1} (x^i - y^i), \tag{6.30}$$

where C^i is the estimated variance of the forecast at day i. If x^i is D-dimensional, then C^i should be a $D \times D$ matrix. The NEES score should be equivalent to the dimension, D, if the algorithm is consistent. As such, in assessing death forecasts, the desired NEES value is $D \approx 1$. However, there are instances in which estimators generate over-cautious and over-confident estimates. Over-cautious estimates cause the estimate variance to dominate the squared estimation error, resulting in a NEES value of less than 1. Over-confident estimates, which are arguably more damaging in terms of their impact on decision making, alternatively cause the squared estimation error to dominate the estimate variance such that the NEES value is greater than 1.

6.4 Computational Experiments

The statistical model outlined in Section 6.2 is calibrated with the England specific surveillance data of Section 6.2.2. The start and end dates for the simulation period are 17th February 2020 and 31st January 2021, respectively. The start dates in the surveillance data are 24th March 2020. The analysis undertaken in this chapter takes into account four prediction windows. The forecasting windows are defined to be between the dates 10th November 2020 - 17th November 2020, 5th December 2020 - 12th December 2020, 30th December 2020 - 6th January 2021 and 24th January 2021 - 31st January 2021. The different simulations are 25 days apart.

Initial parameter values $1/\phi_{\text{deaths}}$, $1/\phi_{\text{admissions}}$, $1/\phi_{\text{calls}}$, $\rho_{\text{admissions},1}$,..., $\rho_{\text{admissions},K+1}$ and $\rho_{\text{calls},1}$,...,$\rho_{\text{calls},L+1}$ are drawn using Stan's default initialisation procedure. The parameters are selected uniformly between -2 and 2 on the unconstrained parameter space. For parameters α_1, α_2, β_1,..., β_J, d_L, d_I, d_P and ω, the implementation draws uniformly from custom intervals to prevent initialisation failures caused by unrealistic parameter values. A bespoke numerical integrator is used to simulate the transmission model ODEs outlined in Section 6.2.1.

The analysis was run on the University of Liverpool's High-Performance Computer (HPC). Each node has two Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz processors, a total of 40 cores and 384 GB of memory. In the following experiments, six independent Markov Chains each draw 512 samples, with the first 256 discarded as *burn-in*. On average, it takes two hours per Markov Chain for one complete run.

Six smoothness hyperparameters for σ_β, outlined in Table 6.3, were employed to highlight the applicability of using scoring rules to evaluate forecasts. Smaller or larger values of σ_β, that restrict or loosen the random walk prior on the effective contact rate $\beta(t)$, can cause the model to under- or over-fit to the data, respectively.

6.5 Results

Table 6.3 presents the mean scores obtained when forecasting the number of deaths for varying σ_β. The results for each hyperparemeter are averaged over the prediction windows outlined in Section 7.3.1.1. Results for the scoring rules LogS, QS, SphS, RPS, and SES outlined in Table 6.2 are negatively inclined. The greater the value of the smoothness hyperparameter, the better the results. The best score for each scoring rule is highlighted

Scoring Rule	σ_β					
	0.0005	0.001	0.0025	0.005	0.01	0.05
LogS	9.242	7.992	7.632	7.285	**7.011**	7.088
QS	0.004	0.002	0.001	0.001	**-0.001**	0.000
SphS	-0.004	-0.019	-0.024	-0.028	-0.042	**-0.044**
RPS	272.701	127.455	104.965	98.828	60.519	**48.313**
SES	669959	85439	261226	37660	**37607**	44607
NEES	5.454	1.094	1.834	1.850	1.687	**1.009**

Table 6.3: Mean scores, averaged over the four prediction windows outlined in Section 7.3.1.1, for the 7-day forecasts for varying σ_β when simulating from the statistical model. The best scores for each scoring rule are highlighted in bold. For NEES, this is the score closest to 1. For all other rules this is the lowest score.

in bold. The NEES score closest to 1 (1.009) is obtained when using the hyperparameter 0.05.

The highlighted scores in Table 6.2 are observed for the hyperparameters 0.01 and 0.05. The results for the hyperparameters 0.0025 and 0.5 for the different predictions windows are presented in Figure 6.3. In all cases the encapsulation of true deaths by the orange confidence intervals highlight that a larger β parameter is better. In some cases when β is 0.0025, the median sample resides outside the posterior distribution's region of high probability mass, as it does for the Hybrid Rosenbrock distribution [140]. However, with the exception of NEES, none of the scoring rules classify this forecast as over-confident. This highlights NEES as a valuable diagnostic tool that should be used alongside proper scoring rules.

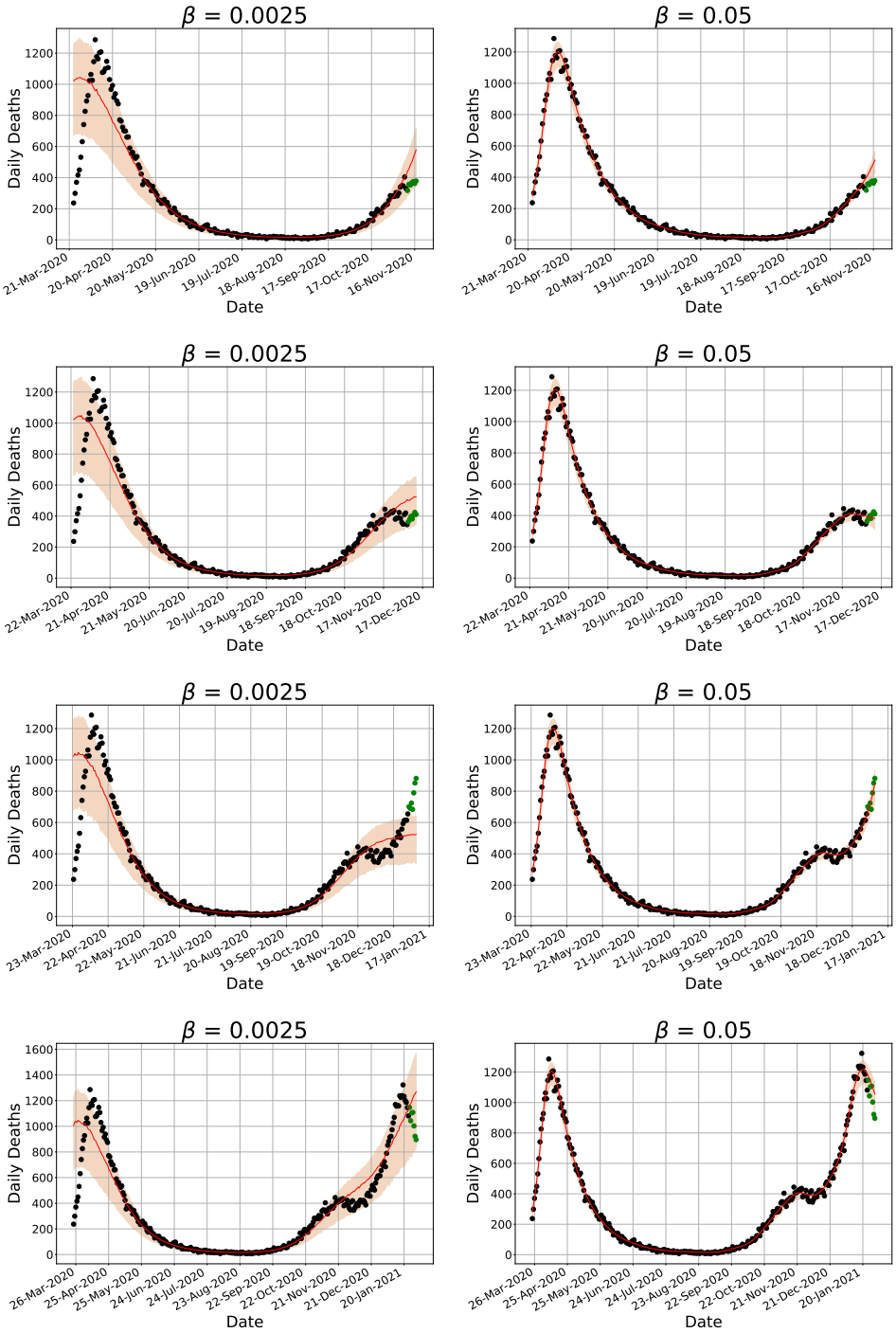


Figure 6.3: Forecasts for the hyperparameters β=0.0025 (left) and β=0.05 (right). Confidence intervals of 1 standard deviation from the mean (orange), the mean sample (red) and the true deaths are given by the black and green dots.

6.6 Conclusions and Future Work

The use of simple scoring rules in helping to detect deficiencies in a statistical model when making forecasts has been outlined in this chapter. The analysis of Section 7.3.1.1 shows that a smaller hyperparameter, σ_β, as parameter estimates allows for more statistical uncertainty to be propagated when forecasting. An example indicating that NEES is an underappreciated scoring rule, due to its ability to detect over-confident and over-cautious forecasts, is provided.

As mentioned in Section 6.1, a limitation of scoring rules is that only forecasts of observable variables can be assessed. One method for evaluating latent quantities, such as the growth rate and reproductive number R_t, is to use simulation based calibration (SBC) [141]. This process involves simulating data from a model using parameters drawn from the prior distribution. The posterior calibration over the independent simulated datasets can then be tested against the inference algorithm.

Extending the statistical model in Section 6.2 to account for age stratification and to simulate the simultaneous spread of a disease in different regions has not yet been considered. This is due to the associated computational effort. One interesting direction for future work is to use a sequential Monte Carlo (SMC) sampler [142] in place of the MCMC sampling algorithm. An example of such a sampler that uses NUTS as the proposal is presented in [111].

Chapter 7

Using Twitter Data to Inform Disease Models

The emergence of the novel coronavirus (COVID-19) in December 2019 in Wuhan City, Hubei Province, China [143] generated a need to quickly and accurately assemble up-to-date information related to its spread. In this chapter, two methods in which Twitter data is useful when modelling the spread of COVID-19 are proposed. Method (1): machine learning algorithms trained in English, Spanish, German, Portuguese and Italian are used to identify symptomatic individuals derived from Twitter. Using the geo-location attached to each tweet, users are mapped to a geographic location to produce a time-series of potential symptomatic individuals. An extended SEIRD epidemiological model is calibrated with combinations of low-latency data feeds, including the symptomatic tweets, and death data to infer the parameters of the model. The usefulness of the data feeds is then evaluated when making predictions of daily deaths in 50 US States, 16 Latin American countries, 2 European countries and 7 NHS (National Health Service) regions in the UK. It is shown that using symptomatic tweets can result in a 6% and 17% increase in mean squared error accuracy, on average, when predicting COVID-19 deaths in US States and the rest of the world, respectively, compared to using solely death data. Method (2): Origin/destination (O/D) matrices, for movements between seven NHS regions, are constructed by determining when a user has tweeted twice in a 24 hour period in two different locations. We show that by increasing and decreasing a social connectivity parameter within an SIR model is found to affect the rate of spread of a disease.

7.1 Introduction

The novel coronavirus (COVID-19) has, at the time of writing, resulted in over 6.88 million deaths and 676 million confirmed cases worldwide [144]. By January 2020, new cases of COVID-19 had been seen throughout Asia, and by the time the World Health Organisation (WHO) declared a global pandemic in March 2020, the disease had spread to over 100 countries. It quickly became imperative to establish reliable data feeds relating to the pandemic, such that researchers and analysts could model the ongoing spread of the disease and inform decision-making by government and public health officials.

The latency and reliability of COVID-19 related data sources can vary. Death data can be seen as reliable when compared with confirmed cases derived from positive test results; however, observations of this data are typically delayed from the initial point of infection. Delays also occur between the occurrence and reporting of deaths. The reliability of confirmed cases is limited as the sampling of those tested varies with time and the reason for testing is often not recorded. In addition, hospital admissions typically occur around 1–2 weeks after infection and so may be considered outdated in relation to the time of initial infection. The extent to which these issues are problematic is likely to vary over time and between countries. For example, reliable, publicly available tests only began to become available a number of months after the outbreak and declaration of the COVID-19 pandemic. As such, information on the spread of the disease was limited and varied between countries. Twitter provides real-time data that overcome the timing limitations of the aforementioned data sources. Correlation between tweets relating to influenza and true influenza counts have been observed in [59, 60, 61]. In addition, it is possible to set up a pipeline for collecting and analysing COVID-19 tweets that can be scaled up to multiple countries in a short amount of time.

Infodemiology and infoveillance [145] refer to the ability to process and analyse data, pertinent to disease outbreaks, that are created and stored digitally in real-time. The availability of these data sets, particularly at the beginning of an outbreak, could provide a noisy but accurate representation of disease dynamics. Prior to the pandemic, tweets relating to influenza-like-illness symptoms were observed to substantially improve predicting capacity [146] and to boost nowcasting accuracy by 13% [147]. Models allowing for early warning detection of multiple diseases are proposed in [147, 148] through analysis of tweet content in real time. Many research papers use social media to gain valuable information relating to the COVID-19 pandemic. Natural language processing (NLP), in particular

determining the sentiment of tweets, is a popular research area. Reference [149] uses sentiment analysis and topic modelling to extract information from conversations relating to COVID-19. When including these data within forecasting models a 48.83–51.38% improvement in predicting COVID-19 cases was observed. Public sentiment relating to COVID-19 prevention measures is analysed in [150]. Depression trends among individuals are analysed in [151]. Emotion was observed to change from fear to anger during the first stages of the pandemic [152]. Misinformation and conspiracy theories propagated rapidly through the Twittersphere [153]. Machine learning algorithms have been used to automatically detect tweets containing self-reported symptoms mentioned by users [154], with [155] observing symptoms reported by Twitter users to be similar to those used in a clinical setting. The analysis in [149, 150, 152, 153, 154] is only conducted with the English language. Analysis conducted in multiple languages is less common. Topic detection and sentiment analysis are conducted in the Portuguese and English language in [156]. Misinformation is detected in English, Hindi and Bengali in [157]. Researchers have yet to use symptomatic tweets in multiple languages to calibrate epidemiological models.

Movement mobility patterns have been derived from anonymised cell phone [158, 159] and Twitter [160, 161] data. Using movement between different geographic locations has been shown to be an effective way of modelling the spread of disease [159, 162, 163, 164]. During an epidemic, limiting the movement of individuals with measures, such as school closures and national lockdowns, can drive the reproduction number below 1 [119]. In Italy, when analysing mobile phone movement data, less rigid lockdown measures led to an insufficient decrease in COVID-19 cases when compared to a more rigid lockdown [165].

This chapter outlines two methods in which Twitter data can be used to help inform disease models. Firstly, using machine learning algorithms trained in multiple languages, symptomatic tweets tweets are detected. A time series of aggregated counts are then used to calibrate an extended SIR compartmental disease model. A comparison is then made with other publicly available data sources when predicting COVID-19 related deaths. Secondly, origin/destination (O/D) matrices are derived from where people tweet in real time. By using a multi-region SIR compartmental disease model, we show that restricting movement between regions can have an effect on the spread of a disease.

7.2 Data Collection

Methods for collecting UK NHS region-specific surveillance data and symptomatic tweets are outlined in Sections 7.2.1 and 7.2.2, respectively. The O/D matrices derived from Twitter mobility are described in Section 7.2.3. Two Twitter API developer credentials were used for data collection, in line with the two objectives of this chapter: (1) querying on COVID-19 keywords and (2) querying on geo-located tweets.

Note that testing methods and criteria for classifying deaths as COVID-19-related may differ between geographic locations. All data sets and the associated code can be found on the CoDatMo GitHub repository [166].

7.2.1 United Kingdom NHS Region-Specific Surveillance Data

Methods for collecting UK NHS region-specific surveillance data are presented in the following subsections. References for obtaining the data are given in Table 7.1. The NHS regions in the UK support local systems and provide more connected and sustainable care for patients through integrated care systems. Every individual born in the UK is entitled to use this public health system.

7.2.1.1 Deaths

Aggregated death counts consist of individuals with COVID-19 as the cause of death on their death certificate or those who died within 60 days of a positive test result.

7.2.1.2 Hospital Admissions

Aggregated admission counts consist of the daily COVID-19 related hospital admissions and the total number of COVID-19 patients.

7.2.1.3 Zoe App

Aggregated Zoe App counts consist of entries of COVID-19 symptoms to a mobile App developed in 2020 to help track COVID-19. Users can input COVID-19 symptoms as well as stating whether they have been tested for COVID-19. The App has since broadened its capacity to track other health related concerns such as cancer and high blood pressure.

Geographic Location	Data Feed	Start Date	Reference
U.S States and the rest of the world	Deaths	24 March 2020	[56]
	Tests	1 March 2020	[56]
	Twitter	13 April 2020	Section 7.2.2
U.K NHS Regions	Deaths	24 March 2020	[133]
	Hospital admissions	19 March 2020	[134]
	Twitter	9 April 2020	Section 7.2.2
	Zoe app	12 May 2020	[167]
	111 calls	18 March 2020	[135]
	111 online	18 March 2020	[135]

Table 7.1: A description of the data feeds used per geographic location, the simulation start dates and references for where the feeds were obtained.

7.2.1.4 111 Calls and 111 Online

Aggregated 111 call and 111 online assessment counts consist of individuals that reported potential COVID-19 symptoms through the NHS Pathways telephone and online assessment services, respectively. The telephone service allows for individuals to speak to a medical specialist regarding health concerns. The 111 online service provides information on where is best to obtain help for the symptoms provided. During the COVID-19 epidemic, both services provided a method for individuals to report COVID-19 symptoms.

7.2.2 Symptomatic Tweets

The geographic locations considered when querying on keywords are:

· **US:** 50 States;

· **Rest of the world:** 2 European and 16 Latin American countries;

· **UK:** 7 NHS regions.

Table 7.1 provides a summary of surveillance data corresponding to each geographical location. Death and positive case data for the US States and the rest of the world (ROW) were downloaded from the dashboard operated by the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) [56].

7.2.2.1 Pre-processing Tweets

Tweepy [168] is the Twitter API written in the programming language Python. The free Twitter streaming API is used in this research, limiting the number of tweets available for download to 1%. The premium API would allow for a higher percentage of tweets to be collected. The API was filtered using 93 keywords in English, German, Italian, Portuguese and Spanish that align with COVID-19 symptoms from the MedDRA database [169]. These terms include those associated with fever, cough and anosmia. The full list of keywords can be found here [166]. While other keywords (e.g., "COVID") were considered, keywords relating to symptoms gave rise to a large number of tweets corresponding to individuals experiencing symptoms. Note that any choice of keywords will inevitably identify tweets that are related to advice or general discussion of the disease. This motivated the use of machine learning to post-process the output from the keyword-based queries, as is discussed further in Section 7.2.2.2.

7.2.2.2 Symptom Classifier Breakdown

A multi-class support vector machine (SVM) [170] is trained with a set of annotated tweets vectorised using a skip-gram model [171]. In the context of categorising tweets, the skip-gram method finds the most related words for a given word by capturing the context and semantic similarity between words. By training the algorithm with tweets relating to COVID-19 symptoms, it will be able to distinguish between the classes below. The annotated tweets are labelled according to the following classes:

1. Unrelated tweet;
2. User currently has symptoms;
3. User had symptoms in the past;
4. Someone else currently has symptoms;
5. Someone else had symptoms in the past.

The total number of tweets that mention symptoms, given by the sum of tweets in classes 2–5, is calculated for each 24 hour period. Geo-tagged tweets are mapped to their

location, e.g. corresponding city, via a series of tests using country-specific shapefiles. Previous studies demonstrate that approximately 1.65% of tweets are geo-tagged [172], where the exact position of the tweeter is recorded using longitude and latitude measurements.

For non-geo-tagged tweets, the author's profile is assessed to ascertain whether they provide an appropriate location. The server is deemed to be offline if any 15 minute period within the previous 24 hours has no recorded tweets. After checking all 96 15 minute periods, the count in each geographical area is multiplied by a correction factor:

$$\text{reported tweet count} = \text{total tweet count} \cdot \frac{96}{96 - \text{downtime periods}}. \tag{7.1}$$

To ensure the labelled tweet data sets used for training and testing are balanced, under- and over-represented classes are randomly up- and down-sampled. A subset of data is used to train the classifier before testing on the remainder. The total number of labelled tweets used for training and testing are provided in Table 7.2. Four metrics outlined in Table 7.2 are used to evaluate the classifier. These include the F1 score, accuracy, precision and recall. True positive (TP) and true negative (TN) classifications are outcomes for which the model correctly predicts positive and negative classes, respectively. Similarly, false positive (FP) and false negative (FN) classifications are outcomes for which the model incorrectly predicts positive and negative classes, respectively. Accuracy, precision, recall and the F1 score, which is the harmonic mean of precision and recall, are given as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{7.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{7.3}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{7.4}$$

$$\text{F1} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}. \tag{7.5}$$

Language	# of Data Used		Performance Measures			
	Training	Testing	F1	Accuracy	Precision	Recall
English	1105	195	0.85	0.85	0.85	0.85
German	412	260	0.89	0.89	0.90	0.89
Italian	254	260	0.97	0.96	0.97	0.96
Portuguese	3507	619	0.77	0.77	0.78	0.80
Spanish	1530	270	0.82	0.85	0.82	0.85

Table 7.2: Testing, training and performance measures for the machine learning classifiers for each language.

7.2.2.3 Comparison of Tweets and Positive Test Results

Figure 7.1 presents a comparison between the classified tweets and confirmed positive test results for five US States and one South American country. Both time-series are standardised between 0 and 1 and have been converted to a 7-day rolling average to smooth out short-term fluctuations. It is evident that, at least in the context of these specific examples, the classified tweets do (by eye) follow the trend of positive test results. In some cases, such as Texas and Chile, there appears to be a lag between tweets and positive test results. This could be caused by a reporting delay in these locations. A more rigorous analyses, such as change point detection, could give a stronger indication of how well the trends in the two time-series match. Note that, for some geographic locations, tweets align less well with the corresponding case counts. This could be caused by issues associated with how cases are recorded in each location or by the processing of the tweets.

7.2.3 Twitter Mobility Origin Destination Matrices

The data collection processes for the derivation of the O/D matrices are now presented.

The flow of individuals travelling from one location to another can be expressed as an $M \times M$ matrix, where M is the number of locations in the simulation area. The observation period of the data is 30 April 2020 to 31 May 2020. England is divided into the seven NHS regions, which are treated as separate locations. Tweets with the geo-location feature are collected using the same framework as described in Section 7.2.2.1; however, different Twitter developer API credentials are used as tweets were not filtered based on keywords. To determine where an individual tweeted, a shapefile containing

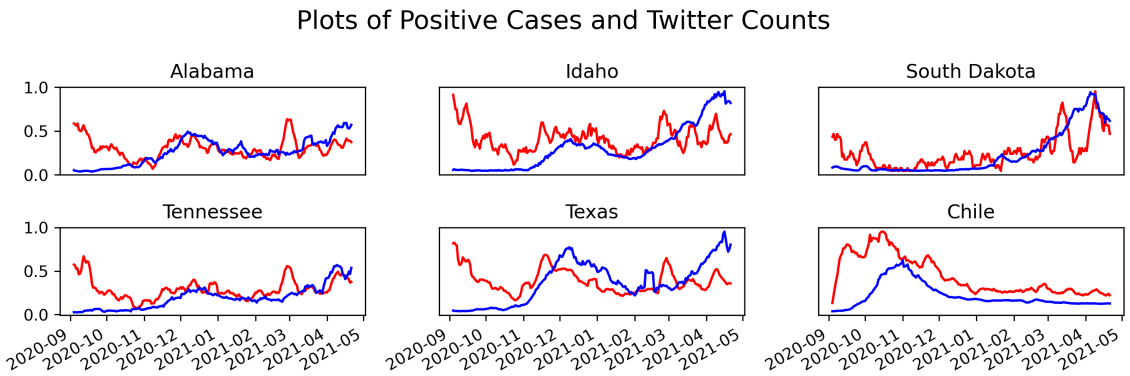


Figure 7.1: Plot of 7-day rolling average and standardised daily counts of positive COVID-19 cases (blue) and self-reported symptomatic tweets (red) for different US States and one South American country.

coordinates of the boundaries of the seven NHS regions is used.

If an individual tweets twice from two distinct locations, for example, London (Origin) and South West (Destination), a movement is subsequently recorded. Figure 7.2 depicts each of these movements in the form of an O/D matrix. Locations on the x- and y-axes represent the origin and destination, respectively. Movements within regions, where an individual tweets multiple times in different locations within the same region, have also been collected. These are observed in the diagonal entries of the matrix.

7.3 Models

The model used for making inferences and death predictions when utilising different data feeds is outlined in Section 7.3.1. The extended SIR disease model catering for movement between different locations is described in Section 7.3.2.

7.3.1 Model for Surveillance Data Comparison

In this analysis, the statistical model outlined in Chapter 6 is used.The observation model (described in Section 6.2.2) outlines the relationship between the transmission model (see Section 6.2.1) and the surveillance data feeds in Table 7.1 during calibration. Daily counts of the surveillance data feeds in Table 7.1 are assumed to follow a negative binomial

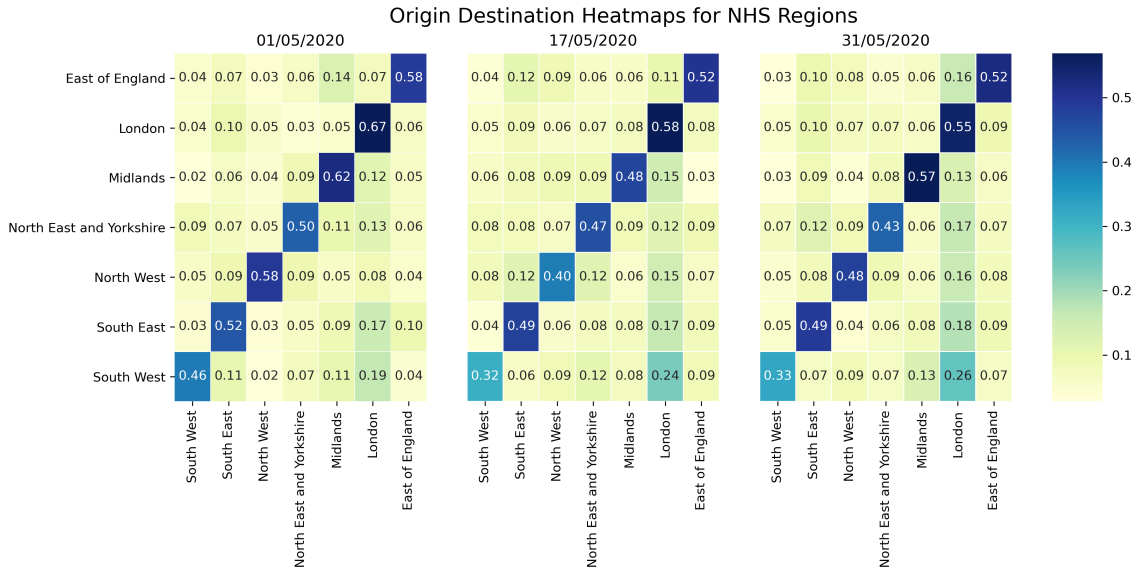


Figure 7.2: Heat-maps of origin destination matrices derived from Twitter for NHS regions. Locations on the x- and y-axes represent the origin and destination, respectively.

distribution parameterised by mean $x(t)$ and over-dispersion parameter ϕ_x, such that

$$x_{\text{obs}}(t) \sim \mathcal{NB}(x(t), \phi_x), \tag{7.6}$$

where x is data feed specific. See Sections 6.2.2.1 and 6.2.2.2 for a description of death and hospital admissions data, respectively.

7.3.1.1 Computational Experiments

The time series considered in the analysis begins on 17 February 2020. The start dates of each data feed follow those outlined in Table 7.1. The terminal time for the US States and the ROW is fixed at 1 February 2021, while, for NHS regions, the terminal time is 7 January 2021. In all cases, forecasts are considered to include seven days.

Similar to the experiments in Chapter 6, the analysis is run on the University of Liverpool's High-Performance Computer (HPC). Each node has two Intel(R) Xeon(R) Gold 6138 CPU @ 2.00 GHz processors, a total of 40 cores and 384 GB of memory. In the following experiments, six independent Markov chains each draw 2000 samples, with the first 1000 discarded as burn-in. Run-time is dependent on the location of the data and the

US States and Rest of the World	NHS Regions
9 July 2020 - 16 July 2020	11 November 2020 - 18 November 2020
17 October 2020 - 24 October 2020	21 November 2020 - 28 November 2020
25 January 2021 - 1 February 2021	1 December 2020 - 8 December 2020
-	11 December 2020 - 18 December 2020
-	21 December 2020 - 28 December 2020
-	31 December 2020 - 7 January 2021

Table 7.3: Prediction windows for the US States and the rest of the world, and NHS regions.

date at which the prediction is made. However, it typically takes 4.5 hours per Markov chain for a complete run.

Initially, the model is only calibrated with death data and forecasts produced for seven daily death counts for the geographic locations described in Section 7.2.2 for the time periods outlined in Table 7.3. These forecasts are set as the baseline when comparing against forecasts incorporating low-latency data feeds.

Two metrics are used to determine the accuracy of the resulting forecasts. First, the mean absolute error (MAE), which shows the average error over a set of predictions, is calculated. This is given by

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x^i - y^i|, \tag{7.7}$$

where N is the number of predictions and x^i and y^i the predicted and true number of deaths on day i, respectively. The percentage difference between forecasts, using only deaths (MAE_D) and those combining deaths with low-latency data feeds (MAE_{DL}), is calculated as follows:

$$\text{MAE \% Diff} = \frac{MAE_{DL} - MAE_D}{MAE_D}, \tag{7.8}$$

where a smaller percentage difference is preferred.

Secondly, the uncertainties associated with the forecasts are considered by assessing the NEES score. A detailed description of this metric is provided in Section 6.3.1.

7.3.2 Model for Utilising Origin Destination Matrices

An extension of the discrete time approximation SIR model that includes movement between geographic locations [159, 173] and is an extension of [174] is now described. The population in location i is denoted P_i. At the beginning of the simulation, P_i is divided into three compartments: susceptible, infected and recovered, denoted $S_{i,t}$, $I_{i,t}$ and $R_{i,t}$, respectively, for timestep t. Note that, since the size of the population is fixed throughout the simulation, subscript t is not required when denoting P_i. Location j represents the set of locations connected to location i. The origin of the pandemic is simulated at a random location, with a fraction of the susceptible compartment infected. The transmission rate in location i on day t is given by $\beta_{i,t}$, while $m_{i,j}$ is the count of individuals travelling from location j to i. The global parameter γ describes the recovery rate.

The proportions of infected and susceptible individuals at location j at time t are denoted $x_{j,t}$ and $y_{j,t}$, respectively. The disease spreads via infected individuals travelling according to the O/D matrices in Figure 7.2. The full extended SIR model is described as follows:

$$S_{i,t+1} = S_{i,t} - \frac{\beta_{i,t} S_{i,t} I_{i,t}}{P_i} - \frac{\alpha S_{i,t} \sum_j m^t_{i,j} x_{j,t} \beta_{j,t}}{P_i + \sum_j m^t_{i,j}}, \tag{7.9}$$

$$I_{i,t+1} = I_{i,t} + \frac{\beta_{i,t} S_{i,t} I_{i,t}}{P_i} + \frac{\alpha S_{i,t} \sum_j m^t_{i,j} x_{j,t} \beta_{j,t}}{P_i + \sum_j m^t_{i,j}} - \gamma I_{i,t}, \tag{7.10}$$

$$R_{i,t+1} = R_{i,t} + \gamma I_{i,t}. \tag{7.11}$$

The number of infected individuals that move from all locations j to location i and transmit the disease to the susceptible population is given by

$$\sum_j m^t_{i,j} x_{j,t} \beta_{j,t}. \tag{7.12}$$

Uninfected individuals at location i are infected by individuals at location j with probability

$$\frac{\alpha S_{i,t} \sum_j m^t_{i,j} x_{j,t} \beta_{j,t}}{P_i + \sum_j m^t_{i,j}}. \tag{7.13}$$

This rate is dependent on α, which describes the intensity of the movement of individuals and is referred to as the social connectivity parameter.

7.4 Results

In this section, results for the two objectives are outlined. Comparison of the accuracy of death forecasts and findings on the impact of movement on the spread of a disease are presented in Sections 7.4.1 and 7.4.2, respectively.

7.4.1 Surveillance Data Comparison

The NEES value and MAE percentage difference between the baseline, ingesting solely deaths, and the incorporation of low-latency data feeds for the US States, the ROW, and NHS regions are given in Tables 7.4–7.6, respectively. For all geographic locations, the results are averaged over the prediction windows described in Table 7.3. A visual representation of these predictions windows can be seen in Figure 7.3.

When ingesting solely death data, tweets, tests, and tweets and tests for US States, the average NEES values are 1.696, 1.409, 1.483 and 1.269, respectively. The corresponding results for the ROW are 0.433, 0.500, 1.198 and 0.723. As explained in Section 7.3.1.1, a NEES value of $\sim$1 is desired, with values $<$1 and $>$1 indicating that the forecast is over-cautious or over-confident, respectively. In both cases, ingesting any combination of the data feeds provides a NEES value closer to 1 than the death only forecast.

MAE results for NHS regions are less consistent. Ingesting hospital admission, 111 call and 111 online data sets provide an average increase in performance of 22%, 17% and 22%, respectively. However, tweets and Zoe App data perform less well, with decreases in performance of 2% and 124%, respectively. This issue may arise because, in these feeds, symptoms are self-diagnosed. Consequently, the counts may include relatively large numbers of individuals who do not have COVID-19.

NEES values for NHS regions when ingesting solely death, hospital admission, tweet, Zoe App, 111 call and 111 online data are 0.662, 0.682, 1.044, 3.160, 0.916 and 0.912, respectively. These results infer that, apart from Zoe App data, where forecasts are overly-confident, ingesting all types of data feeds provides more consistent forecasts. Figure 7.5 exemplifies this finding. In the top image, the forecast encapsulates almost all true deaths. However, when ingesting the Zoe App data, the forecast only encapsulates two out of the seven true deaths, resulting in a NEES value of 6.202, which indicates an over-confident estimate.

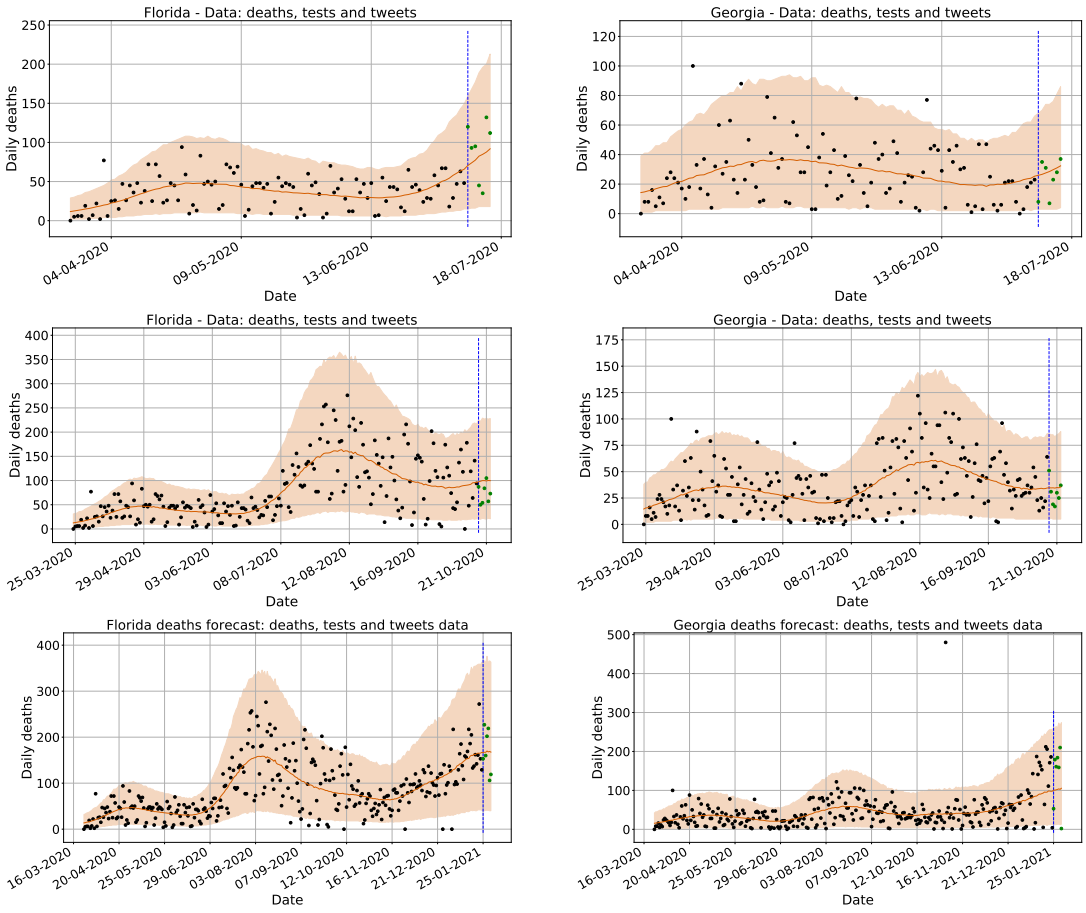


Figure 7.3: Death forecasts in Florida (left) and Georgia (right) using test and tweet data. The first, second and third prediction windows outlined in Table 7.3 are presented in the first, second and third rows, respectively, with confidence intervals of 1 standard deviation from the mean (orange), the mean sample (red) and the beginning of the prediction period (blue). True deaths are given by the black and green dots.

Geographic	Deaths	Tests		Twitter		Tests and Twitter	
Location	NEES	MAE % Diff	NEES	MAE % Diff	NEES	MAE % Diff	NEES
Alaska	0.329	-36	0.334	-29	0.301	-92	0.302
Alabama	0.684	-29	1.874	-29	1.723	-2	1.000
Arkansas	0.275	3	0.317	-1	0.288	-1	0.313
Arizona	0.337	20	0.334	18	0.344	-20	0.244
California	0.611	6	0.709	9	0.802	5	1.206
Colorado	1.886	-25	0.401	-41	0.457	10	1.278
Connecticut	13.406	-8	1.922	-2	0.875	2	11.459
Delaware	3.020	-3	0.918	16	1.046	12	0.727
Florida	0.406	-24	0.179	13	0.353	-20	0.454
Georgia	0.550	9	0.325	41	0.891	-48	0.255
Hawaii	11.459	-12	28.114	-4	24.695	17	10.149
Iowa	19.176	5	7.720	4	1.476	-3	1.600
Idaho	0.914	0	0.809	2	1.791	7	0.986
Illinois	0.573	9	0.350	13	0.319	-116	1.091
Indiana	0.561	-17	0.652	-40	0.781	0	0.481
Kansas	1.021	1	1.037	-2	1.835	1	0.488
Kentucky	0.355	-4	0.374	10	0.548	-15	0.214
Louisiana	0.298	-7	0.305	-2	0.341	9	0.234
Massachusetts	0.351	3	0.342	-3	0.365	14	0.409
Maryland	0.485	-3	0.619	10	0.581	31	0.313
Maine	0.488	1	0.567	-28	0.796	-9	0.952
Michigan	0.592	-6	0.445	-7	0.453	4	0.850
Minnesota	0.683	9	1.019	11	1.200	51	0.747
Missouri	0.810	-7	1.165	-27	1.609	20	0.475
Mississippi	0.683	12	0.721	2	0.997	-15	0.320
Montana	5.034	4	2.244	-1	1.538	-5	5.189
North Carolina	0.908	-1	0.453	9	0.877	-19	0.570
North Dakota	0.513	-32	0.521	-18	0.544	-8	0.661
Nebraska	0.259	5	0.253	7	0.570	5	0.286
New Hampshire	0.252	-74	0.240	-148	0.430	-36	0.288
New Jersey	0.901	-7	0.788	-6	0.926	10	3.177
New Mexico	0.832	-28	0.738	-12	0.969	0	0.489
Nevada	2.129	-24	0.353	-12	0.425	-13	1.904
New York	0.496	31	0.146	3	0.135	-17	0.418
Ohio	0.263	63	0.675	54	0.468	3	0.337
Oklahoma	0.301	-5	0.369	0	0.621	8	0.256
Oregon	0.729	0	1.032	-2	1.692	-4	0.793
Pennsylvania	0.411	-7	0.385	0	0.426	10	0.402
Rhode Island	0.609	-9	0.546	-31	0.446	-2	1.699
South Carolina	2.072	-3	2.157	-4	5.601	-39	0.429
South Dakota	1.259	14	1.080	-2	1.089	2	5.050
Tennessee	0.794	15	1.191	14	1.687	-11	0.600
Texas	0.585	6	0.784	1	0.750	-71	0.706
Utah	0.499	-98	0.716	-127	1.196	13	0.632
Virginia	0.731	-10	0.396	6	0.864	9	0.676
Vermont	0.142	59	0.300	-1	0.163	40	0.043
Washington	0.608	-8	0.561	19	1.787	-1	0.782
Wisconsin	0.842	6	1.028	25	3.921	8	0.850
West Virginia	0.650	-6	0.547	2	1.042	7	0.291
Wyoming	1.939	5	0.951	-15	1.126	25	0.395
Average	**1.696**	**-5**	**1.409**	**-6**	**1.483**	**-5**	**1.269**

Table 7.4: US States: MAE and NEES when using deaths and deaths combined with different low-latency data feeds, averaged over the prediction windows in Table 7.3. Lower MAE % Diff and NEES$\sim$1 = better. Only the English classifier was used.

Geographic Location	Lang	Deaths	Tests		Twitter		Tests and Twitter	
		NEES	MAE % Diff	NEES	MAE % Diff	NEES	MAE % Diff	NEES
Argentina	ES	0.567	3	0.695	−17	0.904	−19	0.765
Bolivia	ES	0.339	−85	0.207	−117	0.182	−118	0.195
Brazil	PT	0.396	−4	0.405	11	0.578	4	0.493
Chile	ES	0.371	15	0.439	14	0.506	10	0.425
Colombia	ES	0.154	17	0.243	−46	0.164	−115	0.223
Costa Rica	ES	0.423	6	0.583	18	3.060	2	0.786
Ecuador	ES	0.156	−26	0.195	−99	0.234	−69	0.234
Guatemala	ES	0.557	−19	0.670	−31	0.815	−31	0.713
Honduras	ES	0.405	−8	0.381	−27	0.915	−41	0.541
Mexico	ES	0.766	16	0.939	11	1.100	11	1.110
Nicaragua	ES	0.091	−13	0.207	−24	1.340	−22	0.364
Panama	ES	0.550	−20	0.421	−4	0.451	−7	0.368
Paraguay	ES	0.535	28	0.877	−7	2.615	8	1.473
Peru	ES	0.507	33	0.103	26	1.630	16	0.515
Uruguay	ES	0.619	11	0.742	−13	0.899	−7	0.643
Venezuela	ES	0.610	−14	0.713	−49	0.890	−91	0.603
Germany	DE	0.379	5	0.613	15	2.131	14	1.570
Italy	IT	0.360	17	0.557	29	3.149	34	1.991
Average		**0.433**	**-6**	**0.500**	**-17**	**1.198**	**-24**	**0.723**

Table 7.5: Rest of the World: MAE and NEES when using deaths and deaths combined with different low-latency data feeds, averaged over the prediction windows in Table 7.3. Lower MAE % Diff and NEES$\sim$1 = better. Language column states which classifier was used.

Geographic	Deaths	Hospital		Twitter		Zoe App		111 Calls		111 Online	
Location	NEES	MAE % Diff	NEES	MAE % Diff	NEES	MAE % Diff	NEES	MAE % Diff	NEES	MAE % Diff	NEES
East of England	0.435	-13	0.419	-7	0.655	38	2.908	-15	0.820	-19	0.795
London	0.878	-36	0.666	-7	1.163	131	3.150	-43	0.750	-47	0.754
Midlands	0.635	-16	0.466	13	0.569	132	3.330	-19	0.418	-47	0.404
North East and Yorkshire	0.753	5	1.188	-4	0.824	153	2.325	-16	0.860	-14	0.888
North West	0.735	-1	0.756	17	1.408	129	3.285	-25	0.932	-25	0.934
South East	0.652	-24	0.805	-3	1.255	126	4.390	8	1.018	6	0.957
South West	0.545	-69	0.474	2	1.432	160	2.729	-8	1.617	-6	1.653
Average	**0.662**	**-22**	**0.682**	**2**	**1.044**	**124**	**3.160**	**-17**	**0.916**	**-22**	**0.912**

Table 7.6: NHS Regions: MAE and NEES when using deaths and deaths combined with different low-latency data feeds, averaged over the prediction windows in Table 7.3. Lower MAE % Diff and NEES$\sim$1 = better. Only the English classifier was used.

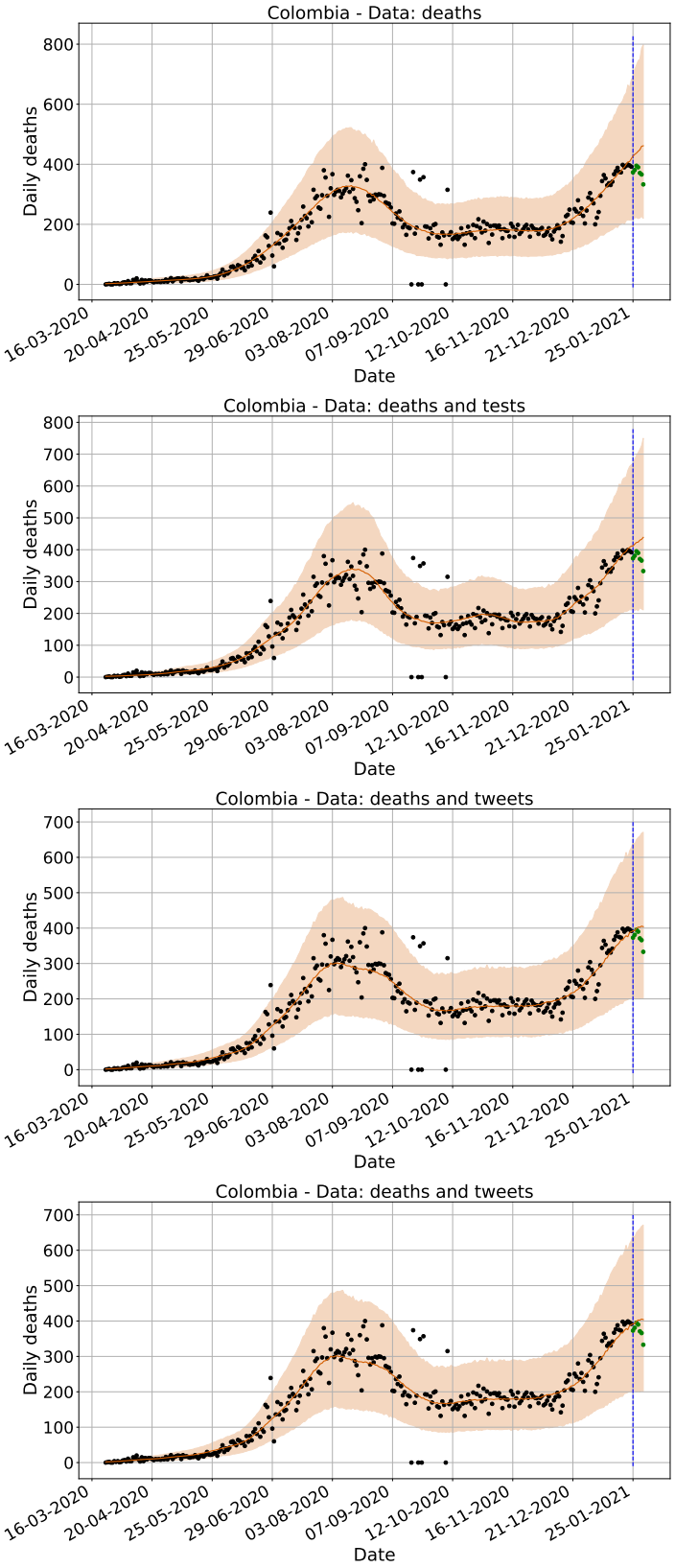
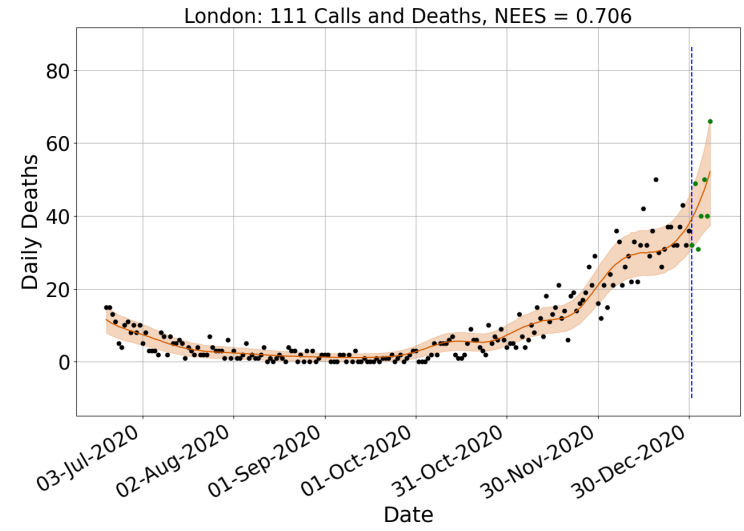
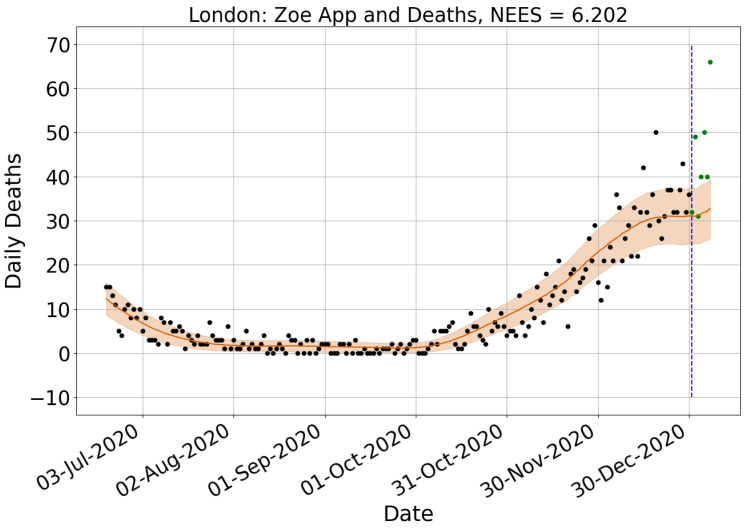


Figure 7.4: Death forecasts in Colombia using combinations of data sets with confidence intervals of 1 standard deviation from the mean (orange), the mean sample (red) and the beginning of the prediction period (blue). True deaths are given by the black and green dots.



(a)



(b)

Figure 7.5: Death forecasts in London using death and 111 call data (top) and Zoe App and death data (bottom) with confidence intervals of 1 standard deviation from the mean (orange), the mean sample (red) and the beginning of the prediction period (blue). The black and green dots are the true deaths.

7.4.2 Origin Destination Matrices Analysis

As explained in Section 7.2.3, a movement is recorded if an individual tweets twice in one day in different locations over a 24 hour period. Counts are assumed to be a percentage of the true population for the seven NHS regions. Figure 7.2 depicts these aggregated movements as O/D matrices.

Figure 7.6 shows the effect of the social connectivity parameter, α, on the spread of a disease and so exemplifies the role of α when simulating disease dynamics. This parameter models the level of contact between individuals when travelling between locations. For example, implementing a lockdown or using a personal car will correspond to increasing values of α. SIR epidemic curves for England are presented on the top row and the infected curves for each NHS region on the bottom row. Limiting contacts within the population through specification of $\alpha = 0.2$ results in the disease ceasing by day 15. For $\alpha = 0.5$, the peak number of infections occurs at approximately day 20 and consists of just over 0.1% of the population. In contrast, when $\alpha = 0.9$, the peak occurs at approximately day 10 and 0.3% of the population are infected. Simulations of the SIR curves under no movement between NHS regions are also provided in the rightmost column of Figure 7.6.

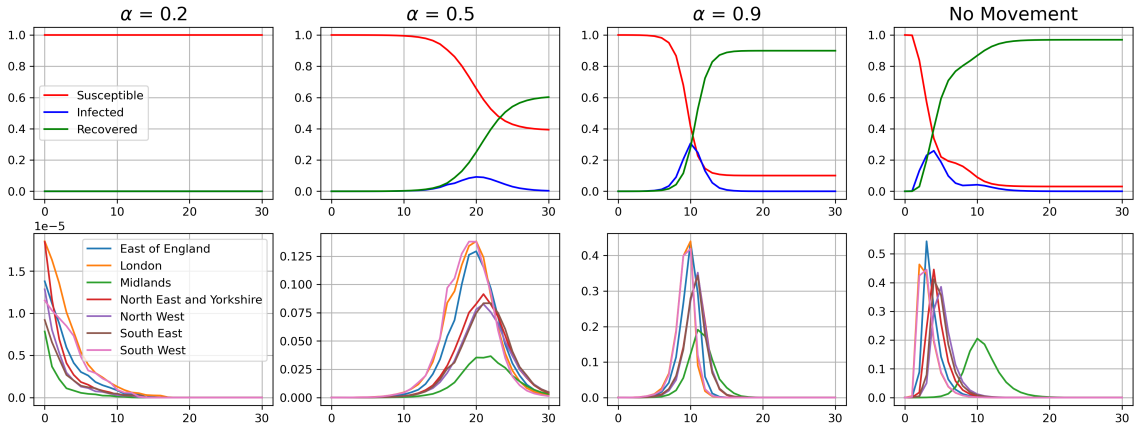


Figure 7.6: Susceptible, infected and recovered epidemic curves for England (top) and infected curves for NHS regions (bottom) for different social connectivity parameters and no movement between regions.

7.5 Conclusions and Future Work

In this chapter, a method for detecting symptomatic COVID-19 tweets in multiple languages has been outlined. Calibrating the epidemiological model outlined in Section 7.3.1 with low-latency data feeds, including symptomatic tweets, was found to provide more accurate and consistent forecasts of daily deaths when compared with using death data alone. A method for extracting movement data from Twitter in the form of O/D matrices was also presented. These movement data were utilised in an extended SIR model to show how a disease originating in one region can quickly spread to others. Restricting movement between regions can be an effective measure when limiting the onward spread of COVID-19.

Incorporating symptomatic tweets for UK regions does not provide the same level of improvement as for other geographic locations. One reason for this reduced improvement could be that daily counts of tweets for NHS regions are less plentiful than for the US States and the ROW. To overcome this, a premium Twitter API that allows the user to download a higher percentage of tweets than that used in this analysis could be purchased. An additional method to increase the hit rate of geo-located tweets is to use natural language processing techniques to estimate the location of the tweet user, such as those outlined in the review [175]. Another direction for future work is to train a more sophisticated

classifier such as the Bidirectional Encoder Representations from Transformers (BERT) classifier [176].

Calibrating the model in Section 7.3.1 with movement data was not explored in this chapter due to the significant computational effort required. One interesting direction for future work would be to use a sequential Monte Carlo (SMC) sampler [142] in place of the MCMC sampling algorithm. An example of such a sampler that uses NUTS as the proposal can be found in [111].

Chapter 8

Conclusions and Future Work

In this thesis, multiple Bayesian methods have been implemented to obtain information pertinent to disease outbreaks. First, a novel method for differentiating the log-likelihood of a particle filter with respect to its parameters was outlined. This advancement enables more accurate inference when estimating the parameters of various SSMs, including stochastic disease models, using p-MCMC, compared to the current state-of-the-art approaches. A framework for differentiating the particle filter using AD and PyTorch as well as the NUTS sampler from PyMC3 was provided.

In addition, an extended SIR compartmental statistical model, implemented in the ppl Stan, was used to model the spread of COVID-19 in England. The high performance MCMC method, NUTS, was used to infer the parameters of the model, nowcast the number of individuals in each compartment and to make forecasts of COVID-19 related deaths. A set of simple scoring rules were employed to evaluate forecasts and detect shortcomings in the statistical model. The scoring rule NEES provides additional information to alternative scoring rules by outlining over-confident and over-cautious forecasts.

The thesis also introduced a novel machine learning framework for detecting COVID-19 related symptoms from tweets. Machine learning algorithms trained in English, Spanish, German, Portuguese and Italian were used to detect symptomatic COVID-19 individuals from the content of their tweets. Symptomatic tweets were aggregated over every 24-hour period, producing daily time series data. Data was obtained for 50 US States, 16 Latin American countries, 2 European countries and 7 NHS regions in the UK. A comparison of calibrating the statistical model with tweets and positive COVID-19 test results was presented for the forecast of death counts. Symptomatic tweets resulted in 6% and 17%

increases in mean squared error accuracy, on average, when predicting COVID-19 deaths in US states and the rest of the world, respectively, compared to using solely death data.

One limitation of the methods outlined in Chapter 4 for differentiating the particle filter with CRN, that has been highlighted in the publication [177], is that the gradient evaluations can be biased (see Figure 4.1). To be unbiased, the average gradient estimate of multiple seeds would need to be taken and the limit considered as the number of seeds tends to infinity. Another method could involve using different seeds at each NUTS iteration (so that the gradients are consistent within a NUTS trajectory but the density function changes between NUTS iterations).

One of the main drawbacks of using p-MCMC and MCMC is the lack of opportunity for parallelisation due to the sequential nature of the two methods. One approach for overcoming this issue is to use an SMC sampler in place of MCMC, with SMC^2 replacing p-MCMC. An advantage of using an SMC sampler and SMC^2 is that they do not have the same constraints as MCMC and p-MCMC. For example, they can bypass the *burn-in* phase and do not need to be reversible.

SMC-Stan, an SMC variant of Stan, is presented in [178]. SMC-Stan allows the user to make inferences on static statistical models described in the Stan programming language by utilising the Stan back end. The original SMC sampler, presented in [142], uses the random walk proposal (see Section 3.3.1) to propose new parameters at each SMC iteration. However, SMC-Stan makes use of HMC and NUTS proposals (see Sections 3.3.2 and 3.3.3, respectively) as presented in [111]. SMC-Stan implements the advancement with approximately optimal L-kernels proposed by [179].

When estimating the weekly reproduction rate R_t for England, the COVID-19 model of Chapter 6 must be continuously calibrated with data from the beginning of the simulation period. As such, assumptions must be made to produce estimates in a timely manner. A parallel implementation of SMC-Stan would improve computation time and thus enable the relaxing of a number of assumptions. One example would be to include age stratification. Computation time is improved through the use of importance sampling. This removes the requirement for the model to be calibrated with data from the beginning of the simulation period each time an estimate is made.

SMC^2 combines two SMC methods: one is an SMC sampler with N_θ particles that makes estimates of the probability density function over the static parameters, θ. The second is a particle filter with N_x particles that makes estimates of the dynamic states. Although significant effort has been made to parallelise the resampling step within the

particle filter [180, 181, 182, 183], the main computational bottleneck associated with SMC^2 is the need to run N_θ particle filters. Future work will therefore involve parallelising the particle filters and the resampling within the SMC sampler. An additional direction for future work will involve using the differentiable particle filter outlined in Chapters 4 and 5 in a parallel implementation of SMC^2.

As described in Chapter 4, methods for obtaining gradients from a particle filter can result in biased or poorly estimated gradients. Therefore, considerations need to be made when using differentiable particle filters within NUTS. One drawback when using gradients that are piecewise continuous is that NUTS's integrator step size adaptation no longer holds. Step size adaption is a calculation that relates the "acceptance statistic" to the associated step size. The step size should be positively correlated with the expectation of the adaptation statistic, such that adjusting the step size should allow for convergence towards the target. However, the presence of a discontinuity may result in the expected value of the adaptation statistic failing to increase as the step size increases. This phenomenon is more pronounced when running models in higher dimensions. Future work will consider the relationship between step size adaption and piecewise continuous gradients in models with higher dimensions. One method to overcome this issue is to use a numerical integrator that can handle discontinuities in log-likelihood and gradient of the log-likelihood [184].

The discrete SIR model presented in Section 3.1.4.2 is commonly used in disease modelling. An example where inferring the parameters using p-MCMC with a MHRW proposal is seen in Section 3.4.1. As the particle state equations in (4.23) follow a discrete distribution and are non-differentiable, calibrating this model using p-MCMC or SMC^2 is restricted to MHRW proposals. Future work will therefore involve incorporating the ideas described in [43, 89, 177] to approximate the discrete operation continuously.

Bibliography

[1] C. C. Kerr, R. M. Stuart, D. Mistry, R. G. Abeysuriya, K. Rosenfeld, G. R. Hart, R. C. Núñez, J. A. Cohen, P. Selvaraj, B. Hagedorn, *et al.*, "Covasim: an agent-based model of covid-19 dynamics and interventions," *PLOS Computational Biology*, vol. 17, no. 7, p. e1009149, 2021.

[2] W. O. Kermack and A. G. McKendrick, "A contribution to the mathematical theory of epidemics," *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, vol. 115, no. 772, pp. 700–721, 1927.

[3] L. K. Whittles and X. Didelot, "Epidemiological analysis of the eyam plague outbreak of 1665–1666," *Proceedings of the Royal Society B: Biological Sciences*, vol. 283, no. 1830, p. 20160618, 2016.

[4] A. Rachah and D. F. Torres, "Predicting and controlling the ebola infection," *Mathematical Methods in the Applied Sciences*, vol. 40, no. 17, pp. 6155–6164, 2017.

[5] D. Osthus, K. S. Hickmann, P. C. Caragea, D. Higdon, and S. Y. Del Valle, "Forecasting seasonal influenza with a state-space sir model," *The annals of applied statistics*, vol. 11, no. 1, p. 202, 2017.

[6] O. Zakary, A. Larrache, M. Rachik, and I. Elmouki, "Effect of awareness programs and travel-blocking operations in the control of hiv/aids outbreaks: a multi-domains sir model," *Advances in Difference Equations*, vol. 2016, no. 1, pp. 1–17, 2016.

[7] Y.-C. Chen, P.-E. Lu, C.-S. Chang, and T.-H. Liu, "A time-dependent sir model for covid-19 with undetectable infected persons," *Ieee transactions on network science and engineering*, vol. 7, no. 4, pp. 3279–3294, 2020.

[8] S. He, Y. Peng, and K. Sun, "Seir modeling of the covid-19 and its dynamics," *Nonlinear dynamics*, vol. 101, no. 3, pp. 1667–1680, 2020.

[9] P. E. Lekone and B. F. Finkenstädt, "Statistical inference in a stochastic epidemic seir model with control intervention: Ebola as a case study," *Biometrics*, vol. 62, no. 4, pp. 1170–1177, 2006.

[10] L. K. Hotta, "Bayesian melding estimation of a stochastic seir model," *Mathematical Population Studies*, vol. 17, no. 2, pp. 101–111, 2010.

[11] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE Proceedings F-radar and signal processing*, vol. 140, pp. 107–113, IET, 1993.

[12] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[13] A. Doucet, A. M. Johansen, *et al.*, "A tutorial on particle filtering and smoothing: Fifteen years later," *Handbook of nonlinear filtering*, vol. 12, no. 656-704, p. 3, 2009.

[14] V. Dukic, H. F. Lopes, and N. G. Polson, "Tracking epidemics with google flu trends data and a state-space seir model," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1410–1426, 2012.

[15] P. Dawson, R. Gailis, and A. Meehan, "Detecting disease outbreaks using a combined bayesian network and particle filter approach," *Journal of theoretical biology*, vol. 370, pp. 171–183, 2015.

[16] D. M. Sheinson, J. Niemi, and W. Meiring, "Comparison of the performance of particle filter algorithms applied to tracking of a disease epidemic," *Mathematical biosciences*, vol. 255, pp. 21–32, 2014.

[17] N. M. Shahtori, C. Scoglio, A. Pourhabib, and F. D. Sahneh, "Sequential monte carlo filtering estimation of ebola progression in west africa," in *2016 American Control Conference (ACC)*, pp. 1277–1282, IEEE, 2016.

[18] H. Abbey, "An examination of the reed-frost theory of epidemics," *Human biology*, vol. 24, no. 3, p. 201, 1952.

[19] K. E. Eilertson, J. Fricks, and M. J. Ferrari, "Estimation and prediction for a mechanistic model of measles transmission using particle filtering and maximum likelihood estimation," *Statistics in Medicine*, vol. 38, no. 21, pp. 4146–4158, 2019.

[20] B. Ristic and P. Dawson, "Real-time forecasting of an epidemic outbreak: Ebola 2014/2015 case study," in *2016 19th International Conference on Information Fusion (FUSION)*, pp. 1983–1990, IEEE, 2016.

[21] C. M. Hazelbag, J. Dushoff, E. M. Dominic, Z. E. Mthombothi, and W. Delva, "Calibration of individual-based models to epidemiological data: A systematic review," *PLoS computational biology*, vol. 16, no. 5, p. e1007893, 2020.

[22] R. G. FitzJohn, E. S. Knock, L. K. Whittles, P. N. Perez-Guzman, S. Bhatia, F. Guntoro, O. J. Watson, C. Whittaker, N. M. Ferguson, A. Cori, *et al.*, "Reproducible parallel inference and simulation of stochastic state space models using odin, dust, and mcstate," *Wellcome Open Research*, vol. 5, 2020.

[23] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics letters B*, vol. 195, no. 2, pp. 216–222, 1987.

[24] R. M. Neal *et al.*, "Mcmc using hamiltonian dynamics," *Handbook of markov chain monte carlo*, vol. 2, no. 11, p. 2, 2011.

[25] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," *arXiv preprint arXiv:1701.02434*, 2017.

[26] M. D. Hoffman, A. Gelman, *et al.*, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo.," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[27] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell, "Stan: A probabilistic programming language," *Journal of statistical software*, vol. 76, no. 1, pp. 1–32, 2017.

[28] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, "Probabilistic programming in python using pymc3," *PeerJ Computer Science*, vol. 2, p. e55, 2016.

[29] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[30] D. A. Rasmussen, O. Ratmann, and K. Koelle, "Inference for nonlinear epidemiological models using genealogies and time series," *PLoS computational biology*, vol. 7, no. 8, p. e1002136, 2011.

[31] A. Wigren, R. S. Risuleo, L. Murray, and F. Lindsten, "Parameter elimination in particle gibbs sampling," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] S. El-Saadani, M. Saleh, and S. A. Ibrahim, "Quantifying non-communicable diseases' burden in egypt using state-space model," *PLOS ONE*, vol. 16, pp. 1–23, 08 2021.

[33] A. A. Koepke, I. M. Longini Jr, M. E. Halloran, J. Wakefield, and V. N. Minin, "Predictive modeling of cholera outbreaks in bangladesh," *The annals of applied statistics*, vol. 10, no. 2, p. 575, 2016.

[34] E. S. Knock, L. K. Whittles, J. A. Lees, P. N. Perez-Guzman, R. Verity, R. G. FitzJohn, K. A. Gaythorpe, N. Imai, W. Hinsley, L. C. Okell, *et al.*, "Key epidemiological drivers and impact of interventions in the 2020 sars-cov-2 epidemic in england," *Science Translational Medicine*, 2021.

[35] J. Dureau, K. Kalogeropoulos, and M. Baguelin, "Capturing the time-varying drivers of an epidemic using stochastic dynamical systems," *Biostatistics*, vol. 14, no. 3, pp. 541–555, 2013.

[36] T. Lux, "Bayesian estimation of agent-based models via adaptive particle markov chain monte carlo," *Computational Economics*, pp. 1–27, 2021.

[37] M. Baguelin, G. F. Medley, E. S. Nightingale, K. M. O'Reilly, E. M. Rees, N. R. Waterlow, and M. Wagner, "Tooling-up for infectious disease transmission modelling," *Epidemics*, vol. 32, p. 100395, 2020.

[38] A. Endo, E. Van Leeuwen, and M. Baguelin, "Introduction to particle markov-chain monte carlo for disease dynamics modellers," *Epidemics*, vol. 29, p. 100363, 2019.

[39] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5101–5108, 2020.

[40] H. Wen, X. Chen, G. Papagiannis, C. Hu, and Y. Li, "End-to-end semi-supervised learning for differentiable particle filters," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5825–5831, IEEE, 2021.

[41] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," *arXiv preprint arXiv:1805.11122*, 2018.

[42] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[43] A. Corenflos, J. Thornton, G. Deligiannidis, and A. Doucet, "Differentiable particle filtering via entropy-regularized optimal transport," in *International Conference on Machine Learning*, pp. 2100–2111, PMLR, 2021.

[44] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Conference on robot learning*, pp. 169–178, PMLR, 2018.

[45] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[46] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[47] C. Rosato, L. Devlin, V. Beraud, P. Horridge, T. B. Schön, and S. Maskell, "Efficient learning of the parameters of non-linear models using differentiable resampling in particle filters," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3676–3692, 2022.

[48] G. Poyiadjis, A. Doucet, and S. S. Singh, "Particle approximations of the score and observed information matrix in state space models with application to parameter estimation," *Biometrika*, vol. 98, no. 1, pp. 65–80, 2011.

[49] P. Del Moral, A. Doucet, and S. S. Singh, "Uniform stability of a particle approximation of the optimal filter derivative," *SIAM Journal on Control and Optimization*, vol. 53, no. 3, pp. 1278–1304, 2015.

[50] C. Nemeth, P. Fearnhead, and L. Mihaylova, "Particle approximations of the score and observed information matrix for parameter estimation in state–space models with linear computational cost," *Journal of Computational and Graphical Statistics*, vol. 25, no. 4, pp. 1138–1157, 2016.

[51] C. Nemeth and P. Fearnhead, "Particle metropolis adjusted langevin algorithms for state space models," *arxiv. org*, 2014.

[52] J. Dahlin, F. Lindsten, and T. B. Schön, "Particle metropolis hastings using langevin dynamics," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6308–6312, IEEE, 2013.

[53] J. Dahlin, F. Lindsten, and T. B. Schön, "Second-order particle mcmc for bayesian parameter inference," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 8656–8661, 2014.

[54] J. Dahlin, F. Lindsten, and T. B. Schön, "Particle metropolis–hastings using gradient and hessian information," *Statistics and computing*, vol. 25, no. 1, pp. 81–92, 2015.

[55] M. Girolami and B. Calderhead, "Riemann manifold langevin and hamiltonian monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 2, pp. 123–214, 2011.

[56] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track covid-19 in real time," *The Lancet infectious diseases*, vol. 20, no. 5, pp. 533–534, 2020.

[57] D. De Angelis, A. M. Presanis, P. J. Birrell, G. S. Tomba, and T. House, "Four key challenges in infectious disease modelling using data from multiple sources," *Epidemics*, vol. 10, pp. 83–87, 2015.

[58] "Coronavirus (covid-19) latest insights: Comparisons." `https://www.ons.gov.uk/peoplepopulationandcommunity/healthandsocialcare/conditionsanddiseases/articles/coronaviruscovid19latestinsights/overview#:~:text=The%20hospital%20admission%20rate%20and,in%20the%20lags%20in%20trends`. Accessed: 1 October 2021.

[59] E. Aramaki, S. Maskawa, and M. Morita, "Twitter catches the flu: detecting influenza epidemics using twitter," in *Proceedings of the 2011 Conference on empirical methods in natural language processing*, pp. 1568–1576, 2011.

[60] A. A. Aslam, M.-H. Tsou, B. H. Spitzberg, L. An, J. M. Gawron, D. K. Gupta, K. M. Peddecord, A. C. Nagel, C. Allen, J.-A. Yang, *et al.*, "The reliability of tweets as a supplementary method of seasonal influenza surveillance," *Journal of medical Internet research*, vol. 16, no. 11, p. e3532, 2014.

[61] D. A. Broniatowski, M. J. Paul, and M. Dredze, "National and local influenza surveillance through twitter: an analysis of the 2012-2013 influenza epidemic," *PloS one*, vol. 8, no. 12, p. e83672, 2013.

[62] C. Rosato, J. Harris, J. Panovska-Griffiths, and S. Maskell, "Inference of stochastic disease transmission models using particle-mcmc and a gradient based proposal," in *2022 25th International Conference on Information Fusion (FUSION)*, pp. 1–8, IEEE, 2022.

[63] R. E. Moore, C. Rosato, and S. Maskell, "Refining epidemiological forecasts with simple scoring rules," *Philosophical Transactions of the Royal Society A*, vol. 380, no. 2233, p. 20210305, 2022.

[64] J. Park, L. Bevan, A. Sanchez-Marroquin, G. Danelian, T. Bayley, H. Manley, V. Bowman, T. Maishman, T. Finnie, A. Charlett, *et al.*, "Combining models to generate a consensus effective reproduction number r for the covid-19 epidemic status in england," *medRxiv*, pp. 2023–02, 2023.

[65] "Reproduction number (r) and growth rate: methodology." `https://www.gov.uk/government/publications/reproduction-number-r-and-growth-rate-methodology/reproduction-number-r-and-growth-rate-methodology`. Accessed: 2022-09-07.

[66] C. Rosato, R. E. Moore, M. Carter, J. Heap, J. Harris, J. Storopoli, and S. Maskell, "Extracting self-reported covid-19 symptom tweets and twitter movement mobility origin/destination matrices to inform disease models," *Information*, vol. 14, no. 3, p. 170, 2023.

[67] A. Doucet, N. De Freitas, N. J. Gordon, *et al.*, *Sequential Monte Carlo methods in practice*, vol. 1. Springer, 2001.

[68] J. Dahlin and T. B. Schön, "Getting started with particle metropolis-hastings for inference in nonlinear dynamical models," *Journal of Statistical Software*, vol. 88, no. 1, pp. 1–41, 2019.

[69] J. Hull and A. White, "The pricing of options on assets with stochastic volatilities," *The journal of finance*, vol. 42, no. 2, pp. 281–300, 1987.

[70] R. Langrock, "Some applications of nonlinear and non-gaussian state–space modelling by means of hidden markov models," *Journal of Applied Statistics*, vol. 38, no. 12, pp. 2955–2970, 2011.

[71] I. Fenemigho, E. Ukponmwan, E. C. Nnakwue, I. Udoete, C. Asuzu, A. Adaralegbe, and U. Effiong, "Covid-19, flattening the curve: recommendations towards control and managing a second wave," *Journal of Global Health Reports*, vol. 4, p. e2020074, 2020.

[72] P. Birrell, J. Blake, E. Van Leeuwen, N. Gent, and D. De Angelis, "Real-time nowcasting and forecasting of covid-19 dynamics in england: the first wave," *Philosophical Transactions of the Royal Society B*, vol. 376, no. 1829, p. 20200279, 2021.

[73] "Negative binomial distribution (alternative parameterization).." `https://mc-stan.org/docs/2_19/functions-reference/nbalt.html#nbalt`. Accessed: 2 January 2023.

[74] D. Lautier, A. Javaheri, and A. Galli, "Filtering in finance," 2003.

[75] W. Yang, A. Karspeck, and J. Shaman, "Comparison of filtering methods for the modeling and retrospective forecasting of influenza epidemics," *PLoS computational biology*, vol. 10, no. 4, p. e1003583, 2014.

[76] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull, "Multiple object tracking using particle filters," in *2006 IEEE Aerospace Conference*, pp. 8–pp, IEEE, 2006.

[77] C. Snyder, "Particle filters, the "optimal" proposal and high-dimensional systems," in *Proceedings of the ECMWF Seminar on Data Assimilation for atmosphere and ocean*, pp. 1–10, 2011.

[78] R. Van Der Merwe, A. Doucet, N. De Freitas, and E. Wan, "The unscented particle filter," *Advances in neural information processing systems*, vol. 13, pp. 584–590, 2000.

[79] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and computing*, vol. 10, no. 3, pp. 197–208, 2000.

[80] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *2006 IEEE nonlinear statistical signal processing workshop*, pp. 79–82, IEEE, 2006.

[81] C. P. Robert and G. Casella, "The metropolis—hastings algorithm," in *Monte Carlo Statistical Methods*, pp. 231–283, Springer, 1999.

[82] D. Van Ravenzwaaij, P. Cassey, and S. D. Brown, "A simple introduction to markov chain monte–carlo sampling," *Psychonomic bulletin & review*, vol. 25, no. 1, pp. 143–154, 2018.

[83] G. O. Roberts and J. S. Rosenthal, "General state space markov chains and mcmc algorithms," *Probability surveys*, vol. 1, pp. 20–71, 2004.

[84] N. Bou-Rabee and J. M. Sanz-Serna, "Geometric integrators and the hamiltonian monte carlo method," *Acta Numerica*, vol. 27, pp. 113–206, 2018.

[85] N. Bou-Rabee and J. M. Sanz-Serna, "Randomized hamiltonian monte carlo," *The Annals of Applied Probability*, vol. 27, no. 4, pp. 2159–2194, 2017.

[86] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical science*, vol. 7, no. 4, pp. 457–472, 1992.

[87] A. Lee, *Towards smooth particle filters for likelihood estimation with multivariate latent variables*. PhD thesis, University of British Columbia, 2008.

[88] J. Elfring, E. Torta, and R. van de Molengraft, "Particle filters: A hands-on tutorial," *Sensors*, vol. 21, no. 2, p. 438, 2021.

[89] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, vol. 33. US Government Printing Office, 1954.

[90] G. Peyré, M. Cuturi, *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[91] A. Ścibior and F. Wood, "Differentiable particle filtering without modifying the forward pass," *arXiv preprint arXiv:2106.10314*, 2021.

[92] D. Foreman-Mackey, "corner. py: Corner plots," *Astrophysics Source Code Library*, pp. ascl–1702, 2017.

[93] A. G. Wills and T. B. Schön, "Stochastic quasi-newton with line-search regularisation," *Automatica*, vol. 127, p. 109503, 2021.

[94] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos, "Smc2: an efficient algorithm for sequential analysis of state space models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 397–426, 2013.

[95] Ö. D. Akyildiz and J. Míguez, "Nudging the particle filter," *Statistics and Computing*, vol. 30, no. 2, pp. 305–330, 2020.

[96] D. Crisan and J. Miguez, "Nested particle filters for online parameter estimation in discrete-time state-space markov models," *Bernoulli*, vol. 24, no. 4A, pp. 3039–3086, 2018.

[97] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[98] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[99] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML 2015 AutoML workshop*, vol. 238, 2015.

[100] H. Ge, K. Xu, and Z. Ghahramani, "Turing: a language for flexible probabilistic inference," in *International conference on artificial intelligence and statistics*, pp. 1682–1690, PMLR, 2018.

[101] "Deep learning frameworks: A survey of tensorflow, torch, theano, caffe, neon, and the ibm machine learning stack." `https://www.microway.com/hpc-tech-tips/deep-learning-frameworks-survey-tensorflow-torch-theano-caffe-neon-ibm-machine-learning-stack/`. Accessed: 1 October 2021.

[102] "convnet-benchmarks." https://github.com/soumith/convnet-benchmarks. Accessed: 1 October 2021.

[103] P. Subramani, "A particle filter method of inference for stochastic differential equations," Master's thesis, University of Waterloo, 2022.

[104] "Using a "black box" likelihood function." https://docs.pymc.io/en/v3/pymc-examples/examples/case_studies/blackbox_external_likelihood.html. Accessed: 2022-11-04.

[105] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[106] M. Betancourt, "Identifying the optimal integration time in hamiltonian monte carlo," *arXiv preprint arXiv:1601.00225*, 2016.

[107] "Hmc algorithm parameters." https://mc-stan.org/docs/2_19/reference-manual/hmc-algorithm-parameters.html. Accessed: 2023-01-02.

[108] S. Ghosh, P. Birrell, and D. De Angelis, "Variational inference for nonlinear ordinary differential equations," in *International Conference on Artificial Intelligence and Statistics*, pp. 2719–2727, PMLR, 2021.

[109] "Diagnosing biased inference with divergences." https://mc-stan.org/users/documentation/case-studies/divergences_and_bias.html. Accessed: 2022-09-07.

[110] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface*, vol. 6, no. 31, pp. 187–202, 2009.

[111] L. Devlin, P. Horridge, P. L. Green, and S. Maskell, "The no-u-turn sampler as a proposal distribution in a sequential monte carlo sampler with a near-optimal l-kernel," *arXiv preprint arXiv:2108.02498*, 2021.

[112] W. C. Roda, M. B. Varughese, D. Han, and M. Y. Li, "Why is it difficult to accurately predict the covid-19 epidemic?," *Infectious disease modelling*, vol. 5, pp. 271–281, 2020.

[113] E. Y. Cramer, E. L. Ray, V. K. Lopez, J. Bracher, A. Brennen, A. J. Castro Rivadeneira, A. Gerding, T. Gneiting, K. H. House, Y. Huang, *et al.*, "Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the united states," *Proceedings of the National Academy of Sciences*, vol. 119, no. 15, p. e2113561119, 2022.

[114] L. Gardner, J. Ratcliff, E. Dong, and A. Katz, "A need for open public data standards and sharing in light of covid-19," *The Lancet Infectious Diseases*, vol. 21, no. 4, p. e80, 2021.

[115] L. P. James, J. A. Salomon, C. O. Buckee, and N. A. Menzies, "The use and misuse of mathematical modeling for infectious disease policymaking: lessons for the covid-19 pandemic," *Medical Decision Making*, vol. 41, no. 4, pp. 379–385, 2021.

[116] S. Purkayastha, R. Bhattacharyya, R. Bhaduri, R. Kundu, X. Gu, M. Salvatore, D. Ray, S. Mishra, and B. Mukherjee, "A comparison of five epidemiological models for transmission of sars-cov-2 in india," *BMC infectious diseases*, vol. 21, no. 1, pp. 1–23, 2021.

[117] T. Maishman, S. Schaap, D. S. Silk, S. J. Nevitt, D. C. Woods, and V. E. Bowman, "Statistical methods used to combine the effective reproduction number, r (t), and other related measures of covid-19 in the uk," *Statistical Methods in Medical Research*, vol. 31, no. 9, pp. 1757–1777, 2022.

[118] M. J. Keeling, L. Dyson, G. Guyver-Fletcher, A. Holmes, M. G. Semple, I. Investigators, M. J. Tildesley, and E. M. Hill, "Fitting to the uk covid-19 outbreak, short-term forecasts and estimating the reproductive number," *Statistical Methods in Medical Research*, p. 09622802211070257, 2022.

[119] S. Flaxman, S. Mishra, A. Gandy, H. J. T. Unwin, T. A. Mellan, H. Coupland, C. Whittaker, H. Zhu, T. Berah, J. W. Eaton, *et al.*, "Estimating the effects of non-pharmaceutical interventions on covid-19 in europe," *Nature*, vol. 584, no. 7820, pp. 257–261, 2020.

[120] C. E. Overton, L. Pellis, H. B. Stage, F. Scarabel, J. Burton, C. Fraser, I. Hall, T. A. House, C. Jewell, A. Nurtay, *et al.*, "Epibeds: Data informed modelling of the covid-19 hospital burden in england," *PLoS computational biology*, vol. 18, no. 9, p. e1010406, 2022.

[121] Q. J. Leclerc, E. S. Nightingale, S. Abbott, and T. Jombart, "Analysis of temporal trends in potential covid-19 cases reported through nhs pathways england," *Scientific reports*, vol. 11, no. 1, pp. 1–8, 2021.

[122] Y. W. Teh, A. Bhoopchand, P. Diggle, B. Elesedy, B. He, M. Hutchinson, U. Paquet, J. Read, N. Tomasev, and S. Zaidi, "Efficient bayesian inference of instantaneous re-production numbers at fine spatial scales, with an application to mapping and nowcasting the covid-19 epidemic in british local authorities," *URL https://rss. org. uk/RSS/media/File-library/News/2021/WhyeBhoopchand. pdfhttps://localcovid. info/2*, vol. 5, no. 6, 2021.

[123] L. Pellis, P. J. Birrell, J. Blake, C. E. Overton, F. Scarabel, H. B. Stage, E. Brooks-Pollock, L. Danon, I. Hall, T. A. House, *et al.*, "Estimation of reproduction numbers in real time: conceptual and statistical challenges," 2022.

[124] C. Czado, T. Gneiting, and L. Held, "Predictive model assessment for count data," *Biometrics*, vol. 65, no. 4, pp. 1254–1261, 2009.

[125] S. Funk, S. Abbott, B. D. Atkins, M. Baguelin, J. K. Baillie, P. Birrell, J. Blake, N. I. Bosse, J. Burton, J. Carruthers, *et al.*, "Short-term forecasts to inform the response to the covid-19 epidemic in the uk," *MedRxiv*, 2020.

[126] K. Sherratt, H. Gruson, R. Grah, H. Johnson, R. Niehus, B. Prasse, F. Sandman, J. Deuschel, D. Wolffram, S. Abbott, *et al.*, "Predictive performance of multi-model ensemble forecasts of covid-19 across european nations," *medRxiv*, pp. 2022–06, 2022.

[127] N. I. Bosse, S. Abbott, A. Cori, E. van Leeuwen, J. Bracher, and S. Funk, "Transformation of forecasts for evaluating predictive performance in an epidemiological context," *medRxiv*, pp. 2023–01, 2023.

[128] A. Chatzilena, E. Van Leeuwen, O. Ratmann, M. Baguelin, and N. Demiris, "Contemporary statistical inference for infectious disease models using stan," *Epidemics*, vol. 29, p. 100367, 2019.

[129] D. Herrera-Esposito and G. de Los Campos, "Age-specific rate of severe and critical sars-cov-2 infections estimated with multi-country seroprevalence studies," *BMC Infectious Diseases*, vol. 22, no. 1, p. 311, 2022.

[130] S. Woody, M. Tec, M. Dahan, K. Gaither, M. Lachmann, S. J. Fox, L. A. Meyers, J. Scott, and U. of Texas at Austin COVID-19 Modeling Consortium, "Projections for first-wave covid-19 deaths across the us using social-distancing measures derived from mobile phones," *Medrxiv*, pp. 2020–04, 2020.

[131] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the nees?: Auto-tuning kalman filters with bayesian optimization," in *2018 21st International Conference on Information Fusion (FUSION)*, pp. 1072–1079, IEEE, 2018.

[132] M. Longbin, S. Xiaoquan, Z. Yiyu, S. Z. Kang, and Y. Bar-Shalom, "Unbiased converted measurements for tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 3, pp. 1023–1027, 1998.

[133] "UK government. 2021 coronavirus (covid-19) in the uk.." `https://coronavirus.data.gov.uk/details/deaths`. Accessed: 1 October 2021.

[134] "UK government. 2021 coronavirus (covid-19) in the uk.." `https://coronavirus.data.gov.uk/details/healthcare`. Accessed: 1 October 2021.

[135] "Potential coronavirus (covid-19) symptoms reported through nhs pathways and 111 online." `https://digital.nhs.uk/data-and-information/publications/statistical/mi-potential-covid-19-symptoms-reported-through-nhs-pathways-and-111-online/latest`. Accessed: 1 October 2021.

[136] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[137] I. J. Good, "Rational decisions," in *Breakthroughs in statistics*, pp. 365–377, Springer, 1992.

[138] W. E. Wecker, "Comment: Assessing the accuracy of time series mode! forecasts of count observations," *Journal of Business & Economic Statistics*, vol. 7, no. 4, pp. 418–419, 1989.

[139] E. S. Epstein, "A scoring system for probability forecasts of ranked categories," *Journal of Applied Meteorology (1962-1982)*, vol. 8, no. 6, pp. 985–987, 1969.

[140] F. Pagani, M. Wiegand, and S. Nadarajah, "An n-dimensional rosenbrock distribution for markov chain monte carlo testing," *Scandinavian Journal of Statistics*, vol. 49, no. 2, pp. 657–680, 2022.

[141] S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman, "Validating bayesian inference algorithms with simulation-based calibration," *arXiv preprint arXiv:1804.06788*, 2018.

[142] P. Del Moral, A. Doucet, and A. Jasra, "Sequential monte carlo samplers," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 3, pp. 411–436, 2006.

[143] "Covid-19: background information." `https://www.gov.uk/government/publications/wuhan-novel-coronavirus-background-information#:~:text=On%2031%20December%202019%2C%20the,City%2C%20Hubei%20Province%2C%20China.` Accessed: 1 October 2021.

[144] "Coronavirus disease 2019." `https://www.worldometers.info/coronavirus/`. Accessed: 3 March 2023.

[145] G. Eysenbach *et al.*, "Infodemiology and infoveillance: framework for an emerging set of public health informatics methods to analyze search, communication and publication behavior on the internet," *Journal of medical Internet research*, vol. 11, no. 1, p. e1157, 2009.

[146] H. Achrekar, A. Gandhe, R. Lazarus, S.-H. Yu, and B. Liu, "Predicting flu trends using twitter data," in *2011 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pp. 702–707, IEEE, 2011.

[147] O. Șerban, N. Thapen, B. Maginnis, C. Hankin, and V. Foot, "Real-time processing of social media with sentinel: A syndromic surveillance system incorporating deep learning for health classification," *Information Processing & Management*, vol. 56, no. 3, pp. 1166–1184, 2019.

[148] L. Espinosa, A. Wijermans, F. Orchard, M. Höhle, T. Czernichow, P. Coletti, L. Hermans, C. Faes, E. Kissling, and T. Mollet, "Epitweetr: Early warning of public health threats using twitter data," *Eurosurveillance*, vol. 27, no. 39, p. 2200177, 2022.

[149] R. Lamsal, A. Harwood, and M. R. Read, "Twitter conversations predict the daily confirmed covid-19 cases," *Applied Soft Computing*, vol. 129, p. 109603, 2022.

[150] R. J. Medford, S. N. Saleh, A. Sumarsono, T. M. Perl, and C. U. Lehmann, "An "infodemic": leveraging high-volume twitter data to understand early public sentiment for the coronavirus disease 2019 outbreak," in *Open Forum Infectious Diseases*, vol. 7, p. ofaa258, Oxford University Press US, 2020.

[151] Y. Zhang, H. Lyu, Y. Liu, X. Zhang, Y. Wang, J. Luo, *et al.*, "Monitoring depression trends on twitter during the covid-19 pandemic: observational study," *JMIR infodemiology*, vol. 1, no. 1, p. e26769, 2021.

[152] M. O. Lwin, J. Lu, A. Sheldenkar, P. J. Schulz, W. Shin, R. Gupta, and Y. Yang, "Global sentiments surrounding the covid-19 pandemic on twitter: analysis of twitter trends," *JMIR public health and surveillance*, vol. 6, no. 2, p. e19447, 2020.

[153] K. Sharma, S. Seo, C. Meng, S. Rambhatla, and Y. Liu, "Covid-19 on social media: Analyzing misinformation in twitter conversations," *arXiv preprint arXiv:2003.12309*, 2020.

[154] M. A. Al-Garadi, Y.-C. Yang, S. Lakamana, and A. Sarker, "A text classification approach for the automatic detection of twitter posts containing self-reported covid-19 symptoms," 2020.

[155] A. Sarker, S. Lakamana, W. Hogg-Bremer, A. Xie, M. A. Al-Garadi, and Y.-C. Yang, "Self-reported covid-19 symptoms on twitter: an analysis and a research resource," *Journal of the American Medical Informatics Association*, vol. 27, no. 8, pp. 1310–1315, 2020.

[156] K. Garcia and L. Berton, "Topic detection and sentiment analysis in twitter content related to covid-19 from brazil and the usa," *Applied soft computing*, vol. 101, p. 107057, 2021.

[157] D. Kar, M. Bhardwaj, S. Samanta, and A. P. Azad, "No rumours please! a multi-indic-lingual approach for covid fake-tweet detection," in *2021 Grace Hopper Celebration India (GHCI)*, pp. 1–5, IEEE, 2021.

[158] H. S. Badr, H. Du, M. Marshall, E. Dong, M. M. Squire, and L. M. Gardner, "Association between mobility patterns and covid-19 transmission in the usa: a mathematical modelling study," *The Lancet Infectious Diseases*, vol. 20, no. 11, pp. 1247–1254, 2020.

[159] R. Goel and R. Sharma, "Mobility based sir model for pandemics-with case study of covid-19," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 110–117, IEEE, 2020.

[160] J. Osorio-Arjona and J. C. García-Palomares, "Social media and urban mobility: Using twitter to calculate home-work travel matrices," *Cities*, vol. 89, pp. 268–280, 2019.

[161] X. Huang, Z. Li, Y. Jiang, X. Li, and D. Porter, "Twitter reveals human mobility dynamics during the covid-19 pandemic," *PloS one*, vol. 15, no. 11, p. e0241957, 2020.

[162] A. Lombardi, N. Amoroso, A. Monaco, S. Tangaro, and R. Bellotti, "Complex network modelling of origin–destination commuting flows for the covid-19 epidemic spread analysis in italian lombardy region," *Applied Sciences*, vol. 11, no. 10, p. 4381, 2021.

[163] S. Gómez, A. Fernández, S. Meloni, and A. Arenas, "Impact of origin-destination information in epidemic spreading," *Scientific reports*, vol. 9, no. 1, pp. 1–9, 2019.

[164] K. Kondo, "Simulating the impacts of interregional mobility restriction on the spatial spread of covid-19 in japan," *Scientific reports*, vol. 11, no. 1, pp. 1–15, 2021.

[165] M. Vinceti, T. Filippini, K. J. Rothman, F. Ferrari, A. Goffi, G. Maffeis, and N. Orsini, "Lockdown timing and efficacy in controlling covid-19 using mobile phone tracking," *EClinicalMedicine*, vol. 25, p. 100457, 2020.

[166] "Codatmo. 2021 welcome to the codatmo site.." `https://codatmo.github.io`. Accessed: 1 October 2021.

[167] "Zoe app: covid-public-data." `https://console.cloud.google.com/storage/browser/covid-public-data;tab=objects?prefix=&forceOnObjectsSortingFiltering=false`. Accessed: 1 October 2021.

[168] J. Roesslein, "tweepy documentation," *Online] http://tweepy. readthedocs. io/en/v3*, vol. 5, p. 724, 2009.

[169] "Covid-19 terms and meddra." `https://www.meddra.org/COVID-19-terms-and-MedDRA`. Accessed: 1 October 2021.

[170] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[171] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[172] K. Leetaru, S. Wang, G. Cao, A. Padmanabhan, and E. Shook, "Mapping the global twitter heartbeat: The geography of twitter," *First Monday*, 2013.

[173] "Modelling the coronavirus epidemic in a city with python.." `https://towardsdatascience.com/modelling-the-coronavirus-epidemic-spreading-in-a-city-with-python-babd14d82fa2`. Accessed: 2 January 2023.

[174] A. Wesolowski, E. zu Erbach-Schoenberg, A. J. Tatem, C. Lourenço, C. Viboud, V. Charu, N. Eagle, K. Engø-Monsen, T. Qureshi, C. O. Buckee, *et al.*, "Multinational patterns of seasonal asymmetry in human movement influence infectious disease dynamics," *Nature communications*, vol. 8, no. 1, pp. 1–9, 2017.

[175] C.-Y. Huang, H. Tong, J. He, and R. Maciejewski, "Location prediction for tweets," *Frontiers in big Data*, vol. 2, p. 5, 2019.

[176] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[177] G. Arya, M. Schauer, F. Schäfer, and C. Rackauckas, "Automatic differentiation of programs with discrete randomness," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10435–10447, 2022.

[178] A. Varsi, *Streaming Multi-core Sample-based Bayesian Analysis*. PhD thesis, University of Liverpool, 2021.

[179] P. L. Green, L. Devlin, R. E. Moore, R. J. Jackson, J. Li, and S. Maskell, "Increasing the efficiency of sequential monte carlo samplers through the use of approximately optimal l-kernels," *Mechanical Systems and Signal Processing*, vol. 162, p. 108028, 2022.

[180] J. Thiyagalingam, L. Kekempanos, and S. Maskell, "Mapreduce particle filtering with exact resampling and deterministic runtime," *EURASIP Journal on Advances in Signal Processing*, vol. 2017, pp. 1–23, 2017.

[181] A. Varsi, L. Kekempanos, J. Thiyagalingam, and S. Maskell, "Parallelising particle filters with deterministic runtime on distributed memory systems," in *IET 3rd International Conference on Intelligent Signal Processing (ISP 2017)*, pp. 1–10, IET, 2017.

[182] A. Varsi, J. Taylor, L. Kekempanos, E. P. Knapp, and S. Maskell, "A fast parallel particle filter for shared memory systems," *IEEE Signal Processing Letters*, vol. 27, pp. 1570–1574, 2020.

[183] A. Varsi, S. Maskell, and P. G. Spirakis, "An o (log2n) fully-balanced resampling algorithm for particle filters on distributed memory architectures," *Algorithms*, vol. 14, no. 12, p. 342, 2021.

[184] A. Nishimura, D. B. Dunson, and J. Lu, "Discontinuous hamiltonian monte carlo for discrete parameters and discontinuous likelihoods," *Biometrika*, vol. 107, no. 2, pp. 365–380, 2020.

Appendices

A Information for Differentiating Kalman Filter

A.1 Partial versus total derivatives

To try to avoid confusion, we use the partial derivative $\partial/\partial\theta$ to mean the derivative by only changing that function argument, and the total derivative $d/d\theta$ to mean also changing

the other arguments depending on it, i.e. if θ is a scalar,

$$\frac{d}{d\theta}f(a(\theta),\theta) \triangleq \lim_{h\to 0}\frac{f(a(\theta+h),\theta+h)-f(a(\theta),\theta)}{h} \tag{1}$$

$$\frac{\partial}{\partial\theta}f(a(\theta),\theta) \triangleq \lim_{h\to 0}\frac{f(a(\theta),\theta+h)-f(a(\theta),\theta)}{h}. \tag{2}$$

A.2 Differentiating a Kalman Filter

We have a transition kernel and a measurement model as follows, where θ is a parameter vector:

$$p(x'|x,\theta) = \mathcal{N}(x';a(x,\theta),Q(x,\theta)) \tag{3}$$

$$p(y|x',\theta) = \mathcal{N}(y;h(x',\theta),R(x',\theta)). \tag{4}$$

Applying an Extended Kalman Filter gives a proposal for x' of the form

$$q(x'|x,\theta,y) = \mathcal{N}(x';\mu(x,\theta,y),C(x,\theta,y)). \tag{5}$$

We wish to calculate the derivatives $\frac{\partial\mu}{\partial x}$, $\frac{\partial\mu}{\partial\theta}$, $\frac{\partial C}{\partial x}$, $\frac{\partial C}{\partial\theta}$. The standard Kalman filter equations are

$$\begin{aligned} S(x,\theta) &= H(a(x,\theta),\theta)Q(x,\theta)H(a(x,\theta),\theta)^T \\ &\quad + R(a(x,\theta),\theta) \end{aligned} \tag{6}$$

$$K(x,\theta) = Q(x,\theta)H(a(x,\theta),\theta)^T S(x,\theta)^{-1} \tag{7}$$

$$\mu(x,\theta,y) = a(x,\theta) + K(x,\theta)(y - h(a(x,\theta),\theta)) \tag{8}$$

$$C(x,\theta) = Q(x,\theta) - K(x,\theta)H(a(x,\theta),\theta)Q(x,\theta) \tag{9}$$

where

$$H(a,\theta) = \left(\frac{\partial h_i}{\partial a_j}(a,\theta)\right)_{ij} \tag{10}$$

is the Jacobian of the measurement function evaluated at the prior mean. We would like to differentiate these with respect to x and θ but the measurement model is defined in terms

of the prior mean $a(x)$. Let

$$\tilde{h}(x,\theta) = h(a(x,\theta),\theta) \tag{11}$$
$$\tilde{H}(x,\theta) = H(a(x,\theta),\theta) \tag{12}$$
$$\tilde{R}(x,\theta) = R(a(x,\theta),\theta). \tag{13}$$

Then

$$S(x,\theta) = \tilde{H}(x,\theta)Q(x,\theta)\tilde{H}(x,\theta)^T + \tilde{R}(x,\theta) \tag{14}$$
$$K(x,\theta) = Q(x,\theta)\tilde{H}(x,\theta)^T S(x,\theta)^{-1} \tag{15}$$
$$\mu(x,\theta,y) = a(x,\theta) + K(x,\theta)(y - \tilde{h}(x,\theta))) \tag{16}$$
$$C(x,\theta) = Q(x,\theta) - K(x,\theta)\tilde{H}(x,\theta)Q(x,\theta). \tag{17}$$

To compute the derivatives of these from the derivatives in a, applying the chain rule gives

$$\frac{\partial\tilde{h}}{\partial x}(x,\theta) = H(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{18}$$
$$\frac{\partial\tilde{h}}{\partial\theta}(x,\theta) = H(a(x,\theta),\theta)\frac{\partial a}{\partial\theta}(x,\theta) + \frac{\partial h}{\partial\theta}(a(x,\theta),\theta) \tag{19}$$
$$\frac{\partial\tilde{R}}{\partial x}(x,\theta) = \frac{\partial R}{\partial a}(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{20}$$
$$\frac{\partial\tilde{R}}{\partial\theta}(x,\theta) = \frac{\partial R}{\partial a}(a(x,\theta),\theta)\frac{\partial a}{\partial\theta}(x,\theta) + \frac{\partial R}{\partial\theta}(a(x,\theta),\theta) \tag{21}$$
$$\frac{\partial\tilde{H}}{\partial x}(x,\theta) = \frac{\partial^2 h}{\partial a^2}(a(x,\theta),\theta)\frac{\partial a}{\partial x}(x,\theta) \tag{22}$$
$$\frac{\partial\tilde{H}}{\partial\theta}(x,\theta) = \frac{\partial^2 h}{\partial a^2}(a(x,\theta),\theta)\frac{\partial a}{\partial\theta}(x,\theta) + \frac{\partial^2 h}{\partial a\partial\theta}(a(x,\theta),\theta) \tag{23}$$

Hence to evaluate the derivatives of (16) and (17), we need

$$a(x,\theta), \frac{\partial a}{\partial x}, \frac{\partial a}{\partial\theta}, Q(x,\theta), \frac{\partial Q}{\partial x}, \frac{\partial Q}{\partial\theta} \tag{24}$$

from the transition model and

$$h(a,\theta), \frac{\partial h}{\partial a}, \frac{\partial h}{\partial\theta}, \frac{\partial^2 h}{\partial a^2}, \frac{\partial^2 h}{\partial a\partial\theta}, R(a(x),\theta), \frac{\partial R}{\partial a}, \frac{\partial R}{\partial\theta} \tag{25}$$

from the measurement model. From this we apply the product rule and the inverse derivative in Appendix B.1.

A.3 Derivatives of multivariate log normal

If

$$\mathcal{N}(x;\mu,C) = \frac{\exp\left(-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)\right)}{\sqrt{|2\pi C|}} \tag{26}$$

then

$$\frac{\partial}{\partial x}\log\mathcal{N} = -C^{-1}(x-\mu) \tag{27}$$

$$\frac{\partial}{\partial \mu}\log\mathcal{N} = C^{-1}(x-\mu) \tag{28}$$

$$\frac{\partial}{\partial C}\log\mathcal{N} = -\frac{1}{2}\left(C^{-1} - C^{-1}(x-\mu)(x-\mu)^T C^{-1}\right). \tag{29}$$

B Matrix derivatives

B.1 Derivative of a matrix inverse

Suppose U is an $N \times N$ invertible matrix with $N \times N$ derivative with respect to θ_r given by $dU/d\theta_r$. Then

$$\frac{\partial(U^{-1})}{\partial\theta_r} = -U^{-1}\left(\frac{\partial U}{\partial\theta_r}\right)U^{-1}. \tag{30}$$

If θ is an R-dimensional vector, $d(U^{-1})/d\theta$ is an $N \times N \times R$ tensor with slice r given by (30).

B.2 Derivative of a matrix square root

Suppose that A is the matrix square root of C, i.e.

$$C = AA. \tag{31}$$

Applying the product rule gives

$$\frac{\partial C}{\partial \theta} = A\frac{\partial A}{\partial \theta} + A\frac{\partial A}{\partial \theta} \tag{32}$$

hence

$$\frac{\partial A}{\partial \theta} = \frac{1}{2}A^{-1}\frac{\partial C}{\partial \theta}. \tag{33}$$

C Code for Particle-NUTS using PyTorch and PyMC3

C.1 Particle filter code

```
def particlefilter(mu, phi, sigmav, y):
    torch.manual_seed(0)
    T = len(y)
    P = 150
    xp, lw, xpNew = torch.zeros(P)
    loglikelihood = torch.zeros(T)
    xp[:] = torch.full((1,P), 0.001)[0]+torch.randn(P)
    lw[:] = -torch.log(torch.ones(1,1)*P)
    eps = Normal(0, 1).rsample(torch.tensor([T, P]))

    for t in range(1,T):
        xpNew = mu + phi * (xp - mu) + sigmav * eps[t]
        newLW = lw + Normal(0, torch.exp(xpNew/2)).log_prob(y[t-1])
        loglikelihood[t] = torch.logsumexp(newLW, dim=0)
        wnorm = torch.exp(newLW-loglikelihood[t])

        neff = 1./torch.sum(wnorm*wnorm)
        if(neff<P/2):
            idx = torch.multinomial(wnorm, P, replacement=True)
            xpNew= xpNew[idx]
            lw[:] = loglikelihood[t] - torch.log(torch.ones(1,1)*P)
        xp = xpNew

    return(loglikelihood[T-2])
```

C.2 Calculating gradients

```
def getLogLikelihoodAndGrad(thetas, y):
    mu    = thetas[0]
    phi   = thetas[1]
    sigmav   = thetas[2]

    mu_     = Variable(torch.tensor(mu),requires_grad=True)
    phi_    = Variable(torch.tensor(phi),requires_grad=True)
    sigmav_ = Variable(torch.tensor(sigmav),requires_grad=True)

    LogLikelihood = particlefilter(mu_, phi_, sigmav_, y)

    LogLikelihood.backward()

    gradLL_mu  = mu_.grad.detach().numpy()
    gradLL_phi = phi_.grad.detach().numpy()
    gradLL_sigmav = sigmav_.grad.detach().numpy()

    LL_           = LogLikelihood.detach().numpy()

    return(np.array([LogLikelihood]),
           np.array([gradLL_mu,
                     gradLL_phi,
                     gradLL_sigmav]))
```

C.3 Log-likelihood with gradient

```
class LogLikeWithGrad(tt.Op):

    itypes = [tt.dvector]
    otypes = [tt.dscalar]

    def __init__(self, obs):
        self.obs = obs
        self.grad_calc = GradTheanoWrapper(self.obs)

    def perform(self, node, inputs, outputs):
        theta, = inputs
        logl, grad = getLogLikelihoodAndGrad(theta, self.obs)

        outputs[0][0] = np.array(logl)[0]

    def grad(self, inputs, outputs):
        theta, = inputs
        grdss = self.grad_calc(theta)
        grad = tt.as_tensor_variable(grdss)
        return [grad]
```

C.4 Declaring PyMC3 Model

```
logl = LogLikeWithGrad(obs)

with pm.Model() as opmodel:

    a = pm.Normal('a', mu=0.0, sigma=1.0)
    b = pm.TruncatedNormal('b', mu=0.95, sigma=0.05)
    c = pm.Gamma('c', alpha=2, beta=10)

    theta = tt.as_tensor_variable([a, b, c])
    pm.Potential(''lik'', logl(theta))
    trace = pm.sample(draws=1000, tune=1000, chains=4)
```