MODEL ASSESSMENT FOR BAYESIAN SPATIO-TEMPORAL EPIDEMIC MODELS FOR COMPLEX DATA SETS USING HYBRID COMPUTATIONAL METHODS

by

David Yew Weng Thong

Submitted for the degree of
Doctor of Philosophy

Department of Actuarial Mathematics and Statistics

School of Mathematical and Computer Sciences

Heriot-Watt University

February, 2019

Abstract

This project investigates the use of model assessment techniques for stochastic spatiotemporal models, with a focus on embedding classical style tests within the Bayesian framework and applying them to study real-world systems. Techniques will be investigated within the context of epidemic models. These models model the spread of a disease, for example, citrus canker, over a spatial region. We will focus on methods of choosing between different transmission kernels. The transmission kernel is a component in the model which determines how the disease spreads over space and time, and is important in choosing the right strategy for the disease, for example, culling of infected individual. The methods for model selection within this context are challenging to develop and implement. Building on recent work within the group which has focused on tests applied to residual processes, we will investigate how likelihood-based tests might be applied to latent processes in order to formulate methods that avoid the sensitivity to parameter priors suffered by purely Bayesian approaches to model comparison. In addition, we extend existing latent residual tests to detect the presence of anisotropic spatial kernels. The power of these tests will be calculated and their advantages and disadvantages investigated, both from a computational and a practical perspective as well from a theoretical perspective. These investigations will be carried out using computational statistical methods performed on simulated and real-world data sets, including the DEFRA data-set for the foot-and-mouth outbreak of 2001. Our investigations show that the likelihood-based methods are able to detect mis-specification of spatial kernel, sometimes exceeding the power of existing latent residual tests. Our directional infection link residual test is shown to be able detect anisotropy in simulated data. Using hybrid computational programming techniques, our tests have been shown to scale to big data sets of 188,361 individuals, and detect mis-specification of kernel in an existing analysis of the data.

Acknowledgements

I would like to thanks my supervisors Professor Gavin J Gibson and Associate Professor George Streftaris at Heriot-Watt University for their guidance and support. I would like to thank Chris Jewell and Hola Kwama Adrakey for their collaboration and contributions. I would like to thank the EPSRC for providing funding for this PhD project.

ACADEMIC REGISTRY
Research Thesis Submission
Please note this form should be bound into the submitted thesis.

Name:	DAVID YEW WENG THONG		
School:	MATHEMATICS AND COMPUTER SCIENCE		
Version: *(i.e. First, Resubmission, Final)*	FINAL	Degree Sought:	PHD

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. Where the thesis contains published outputs under Regulation 6 (9.1.2) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete Inclusion of Published Works Form – see below)
4. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
5. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
6. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
7. Inclusion of published outputs under Regulation 6 (9.1.2) shall not constitute plagiarism.
8. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate:		Date:	

Submission

Submitted By *(name in capitals)*:	DAVID YEW WENG THONG
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by *(name in capitals)*:			
Method of Submission *(Handed in to SSC; posted through internal/external mail)*:			
E-thesis Submitted ***(mandatory for final theses)***			
Signature:		Date:	

ACADEMIC REGISTRY

Inclusion of Published Works

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	
Author 1	
Author 2	
Signature:	
Date:	

Citation details	
Author 1	
Author 2	
Signature:	
Date:	

Citation details	
Author 1	
Author 2	
Signature:	
Date:	

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we focus our attention on specialized tests of goodness-of-fit for spatio-temporal models of infectious disease. It is critical that epidemic models accurately model behaviours that are central for making decisions on control measures, since the wrong choices of control measures could lead to failure to control the epidemic, and significant losses in terms of life, livestock or financial loss (for example, [13, 15]). We propose several new methods for determining model comparison, and extend existing measures for model adequacy. We also develop new computational techniques and algorithms which use the computer's *Graphics Processing Unit* (GPU) to accelerate model fitting and model assessment for epidemic models, increasing the potential for widespread uptake of the methods.

1.1 Definition of infectious disease

An infectious disease is one that is caused by a pathogen, or a toxin produced by such a pathogen [147, 188, 11], that can be spread between *hosts* (members of the population, either single organisms, or multiple organisms e.g. a farm of animals) either directly, for example through coughing or sneezing, or indirectly, for example, through a *vector* (organisms which can transmit the disease from host, or its wastes, to another host or its food or surroundings, for example, blood-sucking insects [147]), or be transmitted through spores dispersed by the wind [147]. The disease may exhibit a *latent period* during which a host may be incubating the pathogen without showing symptoms or yet being able to infect others [55]. An example of an infectious disease is *Citrus Canker*.

Citrus canker is a disease caused by a bacterial pathogen [72, 35]. The disease causes lesions to appear on the fruit, leaves and stems of the citrus plants. This causes the fruit to become unsaleable, and causes disruption in the transport of fruit by triggering movement bans which are commonly put in place as control measures. The bacteria ooze from the lesions, and are transmitted by a combination of wind and rain, where water containing the pathogen is blown to other trees, and enter through the stomata or any existing wounds on the plant, including those caused by pruning and those caused by other organisms. Control measures of citrus canker involve spraying, quarantine and culling of infected trees. Citrus canker has a short latent period of approximately seven to twenty-one days [73], so after apparent eradication, the disease can re-occur [35]. Observe that in this example, there is a spatial, temporal, and stochastic nature to the epidemic. There is also an element of missing data, which is due to the latent period of an infection. This is true for many other epidemics.

1.2 Brief introduction to epidemic models and their evolution

Epidemics have a spatial, temporal and stochastic nature. There have been many epidemic models created for many different purposes sometimes making approximations or ignoring various aspects of epidemics thought to have little impact on results obtained from using such models. The first epidemic models were developed centuries ago. For example, Bernoulli [20] created mathematical models to determine whether mass vaccination for smallpox would increase the overall life expectancy (an early case of using a mathematical model to determine the efficiency of a control measure).

Many modern mathematical models have grown from the work of Kermack and McKendrick [102], which is an “ancestor” of the models used in this thesis.

This model is not used in this thesis, but a brief illustration is provided here, as this model uses many different ideas which are incorporated into the stochastic models used later in this thesis. The Kermack-McKendrick model [102] is a deterministic model which makes the assumption that the overall population size remains constant; that is, the population is closed. Individuals in the population are either susceptible, infected, or removed. The model consists of three differential equations that determine the rates of change of the susceptibles, infectious, and removed numbers of hosts in the population:

$$\begin{aligned}\frac{dS}{dt} &= -\beta SI \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

where $S(t)$, $I(t)$ and $R(t)$ are the proportions of hosts that are susceptible, infected, or removed at time t. The parameter β is the rate of secondary infection. γ is the removal rate ($\frac{1}{\gamma}$ is the average sojourn time in state I). This is known as a *compartmental model* with compartments S, I, R, between which hosts transition.

An interesting result from the Kermack-McKendrick model is known as the epidemic threshold; the conditions for which the number of infectious hosts grows. Observe that $\frac{dI}{dt} > 0 \Leftrightarrow S(t) > \frac{\gamma}{\beta}$. Therefore, if the initial proportion of susceptibles $S(0) < \frac{\gamma}{\beta}$, the epidemic will die out. This result can be expressed in an alternative way: the quantity $R_0 = (\frac{\gamma}{\beta})^{-1} = \frac{\beta}{\gamma}$, known as the basic reproduction number, which can be interpreted as the expected number of infections caused by an infectious host assuming the population of hosts is entirely susceptible. If $S(0) = 1$, the epidemic will grow if $R_0 > 1$ or die out if $R_0 < 1$. There are many extensions to this model, for example, models with more states, which can allow a model to model various phenomena such as immunity, waning immunity, periods where hosts are not infectious but carrying the pathogen. There are also many other modifications that can be made: for example, seasonality, multiple species of host, multiple pathogens, different age classes and so on.

This clear relationship between the parameters and initial conditions, and whether the epidemic “takes off” or “dies out”, is one of the advantages of deterministic models like the Kermack-McKendrick model. In many deterministic models, it may be possible to directly derive an analytical solution for the conditions in which the epidemic spreads and becomes larger, and also analytical results for final size of the infected number of hosts (for example, [102]). In some circumstances, it is possible to solve the differential equations themselves and obtain equations for the proportions of populations of hosts in each state of the model over time (for example, [102, 80]). In addition, deterministic models do not need a large amount of computer power to simulate from, and thus it may be possible to obtain simulations of more complex epidemics than stochastic models. However, despite the strengths of deterministic models, deterministic models cannot answer questions related to probabilities of events happening.

Epidemic models have been developed over time to model the stochastic nature of epidemics, and to model many states of infection, multiple transmission routes (for example, [9, 10, 27, 26, 36, 48, 49, 53, 52, 123, 134, 141, 140, 169]).

Increasing computer power has allowed more advanced models and techniques to be used, and increased the accuracy of predictions. High-dimensional data-sets require large amounts of computer power, which nowadays can only be harnessed through parallel programming techniques [177]. The computer architecture itself has changed from that in the early days of epidemic modelling, requiring new expertise to program.

In the case of epidemic model fitting, the difficulty of the model-fitting process is increased, as a lot of the dynamics underlying the epidemic are unobserved. With spatio-temporal epidemics, the probability of infection depends on the closeness of infectives nearby, but it is often not possible to record the infected hosts because there is a latent period where infected individuals do not show symptoms, or there is a delay between infection and clinical confirmation, creating unobserved data. In this case, the difficulty is often caused by this unobserved data. This will be elaborated upon in further detail in later chapters.

1.3 On the impact of model selection on control measures and policy

Epidemic models are increasingly used as decision-support tools which influence the response to epidemics. Recent examples of epidemics whose control was informed by mathematical modelling are *Clostridium difficile* in Scotland [166], *HIV/AIDS* in India [151], the pandemic response to *Influenza* [87, 101] and *Foot-and-Mouth Disease (FMD)* [49, 48, 100] in the UK. Different assumptions made in the creation of the epidemic model(s) involved in decision-making can result in a very different choice of control strategy [3]. The response to an epidemic may involve changes to government policy, regulation, or the implementation of control measures such as mass culling or vaccination. Therefore, assessment of the validity of the assumptions underlying epidemic models that are used as decision support tools is important for effective decisions regarding the control of the epidemic.

Choosing an appropriate response to an epidemic is a particularly important task because the consequence of a poorly chosen control strategy is the loss of life, or the cost of billions of pounds of financial loss to industry (for example the foot-and-mouth disease epidemic in the UK in 2001 had a heavy cost, both in terms of culled animals, and financial loss [6]). Global trade and travel in the modern world makes the control of disease more important [180, 184]. Sometimes a poorly chosen control strategy may not stop the spread of an epidemic or even result in epidemic spreading at a faster rate (for example, [38]).

1.3.1 A recent example of how epidemic models influenced decision-making

An example of how spatial epidemic models had an influence on epidemic control policy is the response to the 2001 Foot and Mouth disease epidemic (FMD) [6].

The disease spreads in cattle, sheep and pigs, which each have different levels of susceptibility and infectivity [6]. FMD was initially detected at an abattoir and an existing culling policy was put into place (the slaughter of all infected animals on the premises and all "dangerous contacts" with those infected premises) [6]. It was soon realized that the existing policy was not sufficient to contain the epidemic. This was because a large number of infections remained undetected for a period of time before the authorities were notified [6]. At the start of the epidemic, despite requirements to report occurrences of the disease, the outbreak of FMD at Burnside farm, where the disease it thought to have first occurred, was not reported for several weeks. The disease is hard to diagnose in sheep as sheep may show mild or no symptoms, and the symptoms (if they occur) are mild and easily confused with other diseases. It also took time to test whether each case was clinically confirmed.

The Veterinary Laboratories Agency (VLA) used the InterSpread (deterministic) model to make predictions [6]. Model predictions were compared with actual observations to target unexpected cases for further investigation. The Ministry of Agriculture, Fisheries and Food (MAFF) also provided data to four groups of epidemiologists from various universities to analyse. The Imperial College group modelled the epidemic and recommended that a 1 kilometre to 1.5 kilometre call radius should be sufficient to bring the epidemic under control with minimum culling [49]. A contiguous cull was put in place justified by the fact that models by all epidemic modelling teams showed that a contiguous cull would bring the epidemic under control. Approximately 6,000,000 sheep, cattle and pigs were slaughtered to stop the infection from spreading resulting in a cost to the UK public sector of approximately £3 billion and the private sector of approximately £5 billion. If the culling was too aggressive excessive amounts of healthy cows and sheep would have been culled, resulting in further financial losses for farmers and the farming industry. If the culling was not aggressive enough, the epidemic may have continued to spread, incurring a large cost.

1.3.2 The Importance of Testing Model Adequacy and the Importance of the Spatial Kernel in Epidemic Models

In fitting an epidemic model, assumptions need to be made, most of them stemming from the fact that there often is unobserved data. This can arise because some of the transition times of the hosts are unobserved. In some cases, this is because there is a latent period. In other cases, there may be a delay between the time the host becomes infected and the case becomes clinically confirmed as an instance of the disease [182, 6]. To fit a model of how these individuals transition in and out of these unobserved state(s), assumptions need to be made about how the infected individuals pass on the infection to the susceptibles, which is encapsulated in what is called a spatial kernel, a function that models how the infectious challenge from an infected host to a susceptible host falls off with distance. In many cases, the control strategy for an epidemic, for example, culling in the FMD is heavily influenced by this assumption.

Hence, the correct choice of spatial kernel is highly important to creating accurate predictions of an epidemic. The assumption of a specific kernel, contains in itself assumptions about the tailedness of the kernel, the relative susceptibility and infectivity between different population segments and isotropy of the spatial kernel, making this a complex assumption to specify.

Suppose an epidemic with a latent period is being modelled. If the kernel is mis-specified, this leads to incorrect numbers of non-symptomatic infected individuals, which leads to mis-estimation of latent period, which leads to mis-estimation of the infected period (if snapshot data are used). This also leads to mis-estimation of the background levels of infection relative to between-host infection. This leads to incorrect predictions about whether a control strategy would be effective.

This has impact on the control strategy that is chosen: mis-specified kernels lead to ill-advised control strategies. Certain hosts will be seen as more likely to infect other hosts, or be at risk of infection and are more likely to be culled or vaccinated. This can lead to an ineffective culling or immunisation strategy which may fail to control the epidemic, or generate huge financial losses. It could also lead to an entirely unsuitable epidemic control strategy to be performed, for example the purchase of large amounts of vaccine, when an epidemic could die out by itself. Such methods tend to produce other conservative estimates as it is "harder" for a chain to transition to higher dimensional spatial kernels for this reason.

However, the transmission kernel models transitions into a state that is not directly observed, making it difficult to assess the adequacy of the spatial kernel that has been assumed.

1.4 Motivation for developing new epidemic model selection methods

The use of epidemic models in epidemic control makes it necessary to construct tests which are specialized towards detecting specific mis-specification within epidemic models, namely that produced by mis-specification of the transmission kernel, as this is what models the transmission of the pathogen from one host to another. There are existing tests which are used to determine model adequacy, but most approaches are not oriented towards model selection for spatio-temporal epidemic models with unobserved data, and many approaches to epidemic model assessment do not focus on the selection of spatial kernel. The various existing methods for model selection will be reviewed in further detail in later chapters.

1.4.1 Contributions made in this Thesis

In this thesis, we aim to make several contributions to the field of model assessment and comparison in stochastic spatial epidemic models.

First of all, we aim to extend embedded testing methodology [111, 66, 173] for model assessment to model comparison for spatial stochastic compartmental epidemic models, allowing the comparison of two competing models, rather than simply only determining whether there is substantial discrepancy between the model that has been fitted and the data. This thesis investigates the potential for improved detection of model discrepancy that model comparison tests offer, as the comparison of two competing models is a far more focused question than whether a model was adequate or not, and may have the ability to detect discrepancy in certain cases that model adequacy tests may not, for example, in the comparison between two models which are very similar. Two latent likelihood ratio tests will be introduced for this purpose in later chapters, one of them focusing on specific aspects of misfit. The ability to detect discrepancy of these tests will be compared between each other and existing embedded testing methods and their relative merits compared against each other.

Second of all, existing embedded testing methods [111] will be extended to testing for anisotropy in the spatial kernel. As stated earlier, specifying the spatial kernel involves specifying several assumptions, for example, the long or short-tailedness of the spatial kernel, and whether the spatial kernel is isotropic. There are existing methods for the assessment of the tailedness of the spatial kernel. In this thesis, such methods will be extended and modified to test for anisotropy in the spatial kernel. Effectiveness of such methods to detect such mis-specification will be assessed in this thesis.

Third of all, new computational techniques to utilise the graphics processing unit (GPU) as a coprocessor to the CPU on the computer to accelerate the MCMC and model assessment techniques described herein. The algorithms for data augmented MCMC and model assessment techniques will be adapted for rapid calculation on the GPU allowing for potential speed-ups of several hundred times upon conventional implementations. These developments should allow researchers the ability to perform epidemic model fitting and model assessment techniques within a shorter time-frame, allowing accurate results to be obtained when an epidemic is in progress.

1.5 Outline of the rest of the Thesis

The contents for the rest of this thesis is as follows:

Chapter 2 contains a summary of the class epidemic models that will be the focus of this thesis, spatial stochastic compartmental epidemic models, and the approach to inference for such models, comprised of Markov Chain Monte Carlo and Reversible Jump Markov Chain Monte Carlo methods. This chapter outlines the data augmentation techniques that are commonly used for inference in the presence of unobserved infected hosts in an epidemic. *Chapter 3* contains a review of model assessment methods commonly in use, and the drawbacks of existing methods. In particular a review of embedded testing methods and their theoretical background will be detailed here, and their interpretation. This thesis extends upon existing embedded testing methods for spatial stochastic compartmental epidemic models. These innovations will be described in this chapter. *Chapter 4* describes the comparison of these novel testing methods with existing methods using simulated data. *Chapter 5* details the background theory and innovations involved in using the graphics processing unit (GPU) as a coprocessor to the CPU to accelerate the methods described herein. *Chapter 6* assesses the ability of the novel testing methods for the detection of anisotropy to detect discrepancy in simulated data between a model with an isotropic spatial kernel and the simulated data set in which an anisotropic kernel was utilised for generating the data. *Chapter 7* demonstrates how these previous innovations can be applied to a large and complex dataset, namely the foot-and-mouth disease data-set of 2001, in which a combination of the testing methods mentioned previously and the GPU computation methods are used to rapidly assess the fit of a model to the data. *Chapter 8* concludes the thesis with a review of the conclusions that have been obtained from the previous chapters.

Chapter 2

Epidemic models

In the first chapter, we have described infectious disease epidemics, epidemic models and motivated the importance of model selection for designing control measures.

In this chapter, we will go into further depth, formally introducing the class of models that will be the focus of this thesis and giving a more technical account of their nature. We will also describe the techniques that will be used to estimate parameters in these models and present some other technical material that will be used in the course of the thesis.

2.1 Description of the epidemic model

The model that we will be investigating in this thesis is the spatio-temporal SEIR model (Susceptible-Exposed-Infectious-Removed), as described in [63]. This is a compartmental model in which hosts move between four states: Susceptible (S), Exposed (E), Infectious (I) and Removed (R). Let $S(t)$, $E(t)$, $I(t)$, $R(t)$ be the set of hosts in the relevant state at time t. Hosts transition from $S \to E$, $E \to I$ and $I \to R$. Similar population structures have been used in models in [36] and [96] for foot and mouth disease, and for citrus canker, for example in [38] and [134]. The hosts (which are indexed by $1, 2, \ldots, N$) are distributed over a 2-dimensional region at known points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where the population is of a known size N. It is useful to identify an individual i with its location $\mathbf{x}_i$ without ambiguity. Hosts in state S at time t, in the set $S(t)$ experience infectious challenge from two sources: primary infectious challenge, from sources/sites external to the system under study, and secondary infectious challenge, from infectious individuals within the system. Let α and β be the primary and secondary infection rates. Then the probability of exposure (for an arbitrary host $j \in S(t)$) can be modelled by the following equation:

$$\begin{aligned}\Pr(j \text{ exposed during } [t, t+dt]) &= C(\boldsymbol{x}_j, t)\, dt + o(dt) \\ &= \left(\alpha + \beta \sum_{i \in I(t)} K(\boldsymbol{x}_j, \boldsymbol{x}_i, \kappa)\right) dt + o(dt)\end{aligned} \tag{2.1.1}$$

The function $K(\boldsymbol{x}, \boldsymbol{y}, \kappa)$ is known as the transmission kernel, which models the effect of locations on the infectious challenge from each infectious host to $\mathbf{x}$. Common transmission kernels specified are the *exponential kernel* $K(\boldsymbol{x}, \boldsymbol{y}, \kappa) = \exp\{-\kappa|\boldsymbol{x} - \boldsymbol{y}|\}$ and the *power-law kernel* $K(\boldsymbol{x}, \boldsymbol{y}, \kappa) = (1 + |\boldsymbol{x} - \boldsymbol{y}|^{\kappa})^{-1}$ (used for example in [123, 134, 36, 38]) where $|\cdot|$ is the Euclidean distance. Other distance norms can be used, for example, to take into account population structures such as households, schools etc. (for example, [104, 169, 28, 171]. Other effects can be incorporated into the transmission kernel, such as different levels of infectivity and susceptibility between hosts. The sojourn times in E and I are usually modelled as being independent of the time of entry into the current state, usually with a Gamma, Weibull or Exponential distribution (for example, [145, 100, 97, 96, 134, 131]).

Readers should note that a host may be an individual (for example, [134, 145]), or several individuals, for example a farm or site (for example, [71, 96, 100]). In addition, effects can be included into the models such as network-based transmission without much difficulty (for example,[95]).

2.2 Simulating Data from the Epidemic Model

It is crucial to be able to generate simulated data from the model. These data will be used to test and implement the methods for model selection detailed later in this thesis. Here we are using a variant of the Gillespie algorithm (as in [63]).

Suppose that we have a population of size N, with members situated at points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. Let the probability densities of the sojourn times of hosts in the E state and I state, be f_E and f_I respectively.

1. Set $t = 0$. Set $S(0) = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ (all members are in state S) and $E(0) = I(0) = R(0) = \varnothing$.

2. Set t'_{min} to be the minimum sojourn times of the hosts in sets $E(t)$ and $I(t)$. Let this host that this sojourn time belongs to be denoted k. If $E(t) = \emptyset$ and $I(t) = \emptyset$ then $t'_{min} = \infty$.

3. Draw the waiting time τ for the next exposure from its distribution $\tau \sim \text{Exp}(\sum_{i \in S(t)} C(x_i, t))$.

4. If $t + \tau < t'_{min}$the next transition is from S to E. Draw a point, $\mathbf{j} \in S(t)$ from all the points in $S(t)$ with probability $\frac{C(x_j,t)}{\sum_{i \in S(t)} C(x_i,t)}$. This is the host that will transition from state S to E at time $t + \tau$. Generate a sojourn time for this host from the sojourn time distribution f_E.

5. If $t + \tau > t'_{min}$the next transition is from E to I or I to R. The next transition is host k which transitions into the next state and is moved into sets $I(t)$ or $R(t)$ accordingly. If the host transitions from E to I, generate a sojourn time for this host from the sojourn time distribution f_I.

6. Repeat from step 2 onwards, until the desired stage in the simulation is reached.

2.2.1 An Example Data-set

To illustrate the dynamics of an epidemic produced by the SEIR epidemic model, several simulated datasets have been generated with the above algorithm. The epidemic simulations used similar parameters to those found in [145] where a similar model was used to model a Huanglongbing epidemic in citrus fruit in Clewiston, Florida. The parameters were based on posterior distributions obtained in the paper for trees with ages in the range of 3 to 10 years old. The primary infection, secondary infection and kernel parameters used to generate the data were $\alpha = 0.075\,\text{yr}^{-1}, \beta = 25.0\,\text{yr}^{-1}, \kappa = 0.2$. The sojourn times in the E state were Gamma distributed with shape 14.0 and scale 0.125 yr. The sojourn times in the I state were Gamma distributed with shape 100.0 and scale 0.002 yr. The simulated data-sets contained 1000 hosts uniformly distributed across a square shaped region. Figures 2.2.1 and 2.2.2 show the epidemics generated with the same parameters, with an exponential transmission kernel in fig. 2.2.1 and a power-law transmission kernel in fig. 2.2.2. Note the "patchy" spread of the epidemic under a power law kernel compared to the clustering of exposures next to infectious hosts with the exponential kernel.

2.2.2 Likelihood Function

Key to estimating parameters based on observation of an epidemic will be the likelihood function, as the methods of inference detailed here all take the likelihood as a starting point, and thus follow the likelihood principle [17, 23, 92]. In this models used in this thesis, the transition times from the exposed state to the infectious state and from the infectious state to the removed state were assumed to have independent Gamma probability densities $f_E(\cdot; \alpha_E, \nu_E)$ and $f_I(\cdot; \alpha_I, \nu_I)$, where

$$f(x; \alpha, \beta) = \frac{1}{\Gamma(\alpha)\nu^{\alpha}} x^{\alpha-1} \exp\left(-\frac{x}{\nu}\right)$$

Let the corresponding cumulative distribution functions for the sojourn times be denoted $F_E(\cdot; \alpha_E, \nu_E)$ and $F_I(\cdot; \alpha_I, \nu_I)$. We shall assume that the times that the ith host entered the exposed, infectious and removed state are observed values $t_E^{(i)}, t_I^{(i)}, t_R^{(i)}$. Let the full set of event times for a given realisation be $\mathbf{x} = (\mathbf{t}_E, \mathbf{t}_I, \mathbf{t}_R)$. If a host does not transition during the period in which observations are recorded, the transition time is set to ∞. Let the observation period be denoted as the time

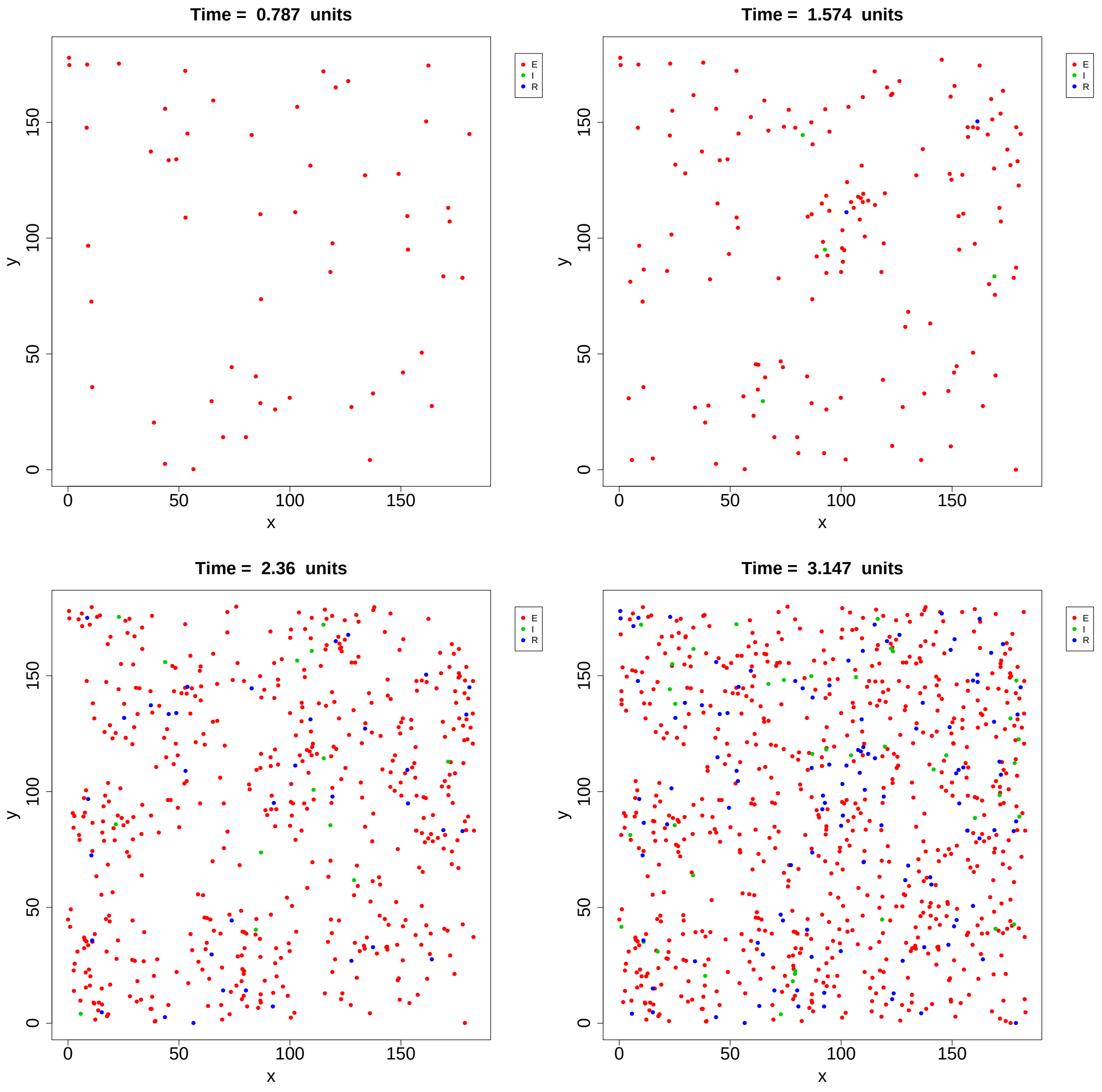

Figure 2.2.1: Snapshots of the simulated epidemic generated with an exponential kernel. Each point on the graph represents one host. Points are colour-coded to represent the current state of the host. Susceptible points are not displayed to maintain clarity of the graph. The colour of the points on the graph indicate the state of each host at the given time. Red indicates the host is exposed, green indicates the host is infectious and blue indicates that the host is removed.

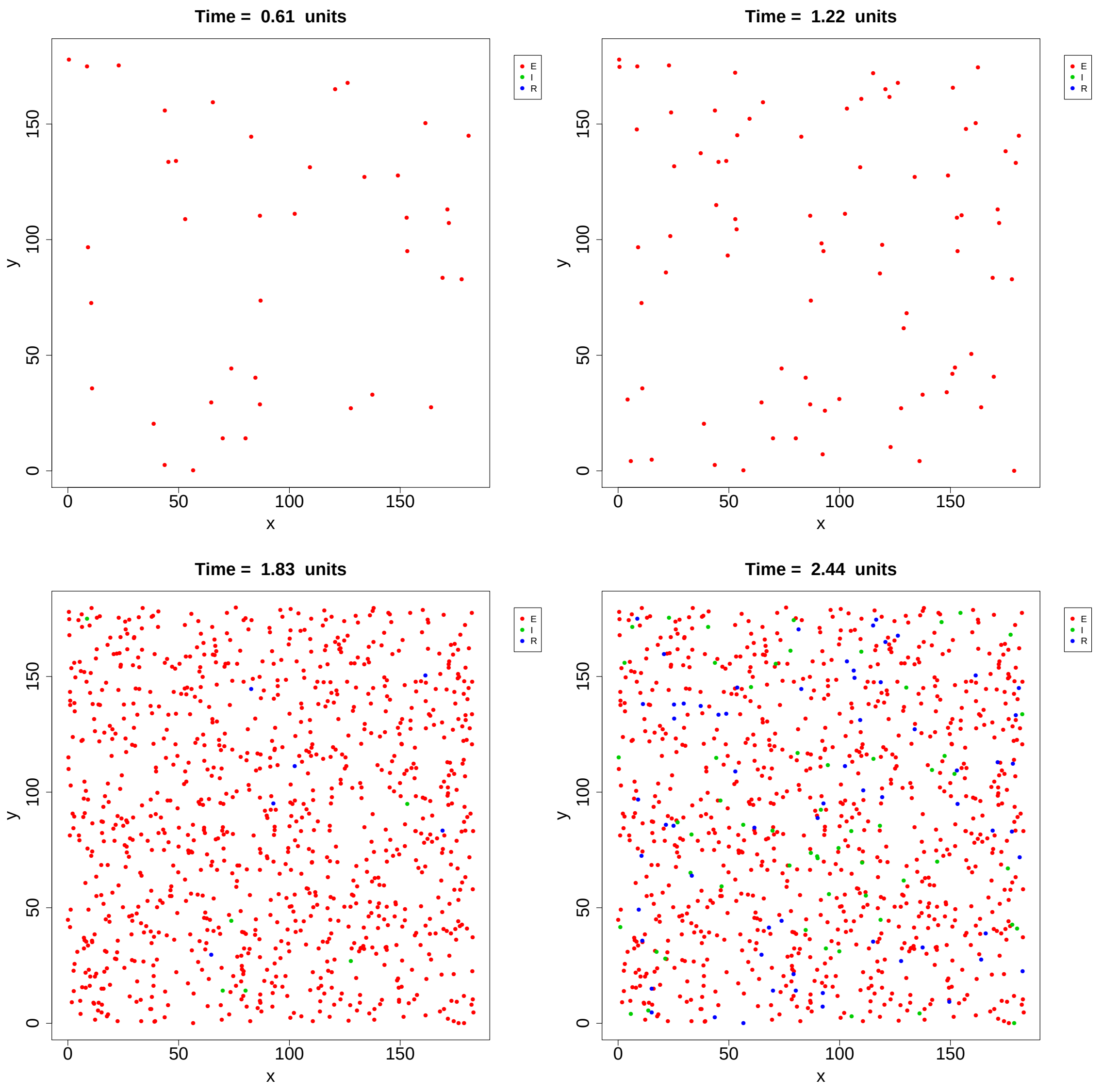

Figure 2.2.2: Snapshots of the simulated epidemic generated with a power-law kernel. Each point on the graph represents one host. Points are colour-coded to represent the current state of the host. Susceptible points are not displayed to maintain clarity of the graph. The colour of the points on the graph indicate the state of each host at the given time. Red indicates the host is exposed, green indicates the host is infectious and blue indicates that the host is removed.

interval $(0, T)$. The likelihood function can be derived as follows (derivation is similar to [36]):

$$\begin{aligned} L\left(\alpha, \beta, \kappa, \alpha_E, \nu_E, \alpha_I, \nu_I; \mathbf{x}\right) &= L\left(\alpha, \beta, \kappa; \mathbf{t}_E\right) \cdot L\left(\alpha_E, \nu_E; (\mathbf{t}_I - \mathbf{t}_E)\right) \cdot L\left(\alpha_I, \nu_I; (\mathbf{t}_R - \mathbf{t}_I)\right) \\ &= \prod_i L\left(\alpha, \beta, \kappa; \mathbf{t}_E^{(i)}\right) \cdot L\left(\alpha_E, \nu_E; (\mathbf{t}_I^{(i)} - \mathbf{t}_E^{(i)})\right) \cdot L\left(\alpha_I, \nu_I; (\mathbf{t}_R^{(i)} - \mathbf{t}_I^{(i)})\right) \end{aligned} \tag{2.2.1}$$

As by (2.1.1) in Section 2.1 on page 11,

$$\Pr(i \text{ exposed during } [t, t + dt]) = C(x_i, t)\, dt + o(dt)$$

the transitions times from S to E have hazard rate at time t, and hence:

$$L\left(\alpha, \beta, \kappa; \mathbf{t}_E^{(i)}\right) = \begin{cases} \exp\left[-\int_0^{t_E^{(i)}} C(x_i, t)\, dt\right] \cdot C(x_i, t_E^{(i)}) & t_E^{(i)} \leq T \\ \exp\left[-\int_0^{T} C(x_i, t)\, dt\right] & t_E^{(i)} > T \end{cases} \tag{2.2.2}$$

using the relationship between hazard rate, probability density function and survival function. Also, using the sojourn time densities and cumulative distribution functions,

$$L\left(\alpha_E, \nu_E; (\mathbf{t}_I^{(i)} - \mathbf{t}_E^{(i)})\right) = \begin{cases} f_E(t_I^{(i)} - t_E^{(i)}; \alpha_E, \nu_E) & t_I^{(i)} \leq T \\ 1 - F_E(T - t_E^{(i)}; \alpha_E, \nu_E) & t_I^{(i)} > T > t_E^{(i)} \end{cases} \tag{2.2.3}$$

and,

$$L\left(\alpha_I, \nu_I; (\mathbf{t}_R^{(i)} - \mathbf{t}_I^{(i)})\right) = \begin{cases} f_I(t_R^{(i)} - t_I^{(i)}; \alpha_I, \nu_I) & t_R^{(i)} \leq T \\ 1 - F_I(T - t_I^{(i)}; \alpha_I, \nu_I) & t_R^{(i)} > T > t_I^{(i)} \end{cases} \tag{2.2.4}$$

and hence, by substituting (2.2.2), (2.2.3), (2.2.4) into (2.2.1):

$$\begin{aligned}
L(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) &\propto \left(\prod_{i;t_E^{(i)}\le T} \exp\left[-\int_0^{t_E^{(i)}} C(x_i,t)\,dt\right]\cdot C(x_i,t_E^{(i)})\right)\cdot \qquad (2.2.5)\\
&\left(\prod_{i;t_E^{(i)}>T} \exp\left[-\int_0^{T} C(x_i,t)\,dt\right]\right)\cdot\left(\prod_{i;t_I^{(i)}\le T} f_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right)\cdot\\
&\left(\prod_{i;t_I^{(i)}>T}\left(1-F_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right)\right)\cdot\left(\prod_{i;t_R^{(i)}\le T} f_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right)\cdot\\
&\left(\prod_{i;t_R^{(i)}>T}\left(1-F_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right)\right)\\
&= L_{SE}(\alpha,\beta,\kappa;\mathbf{x})\cdot L_{EI}(\alpha_E,\nu_E;\mathbf{x})\cdot L_{IR}(\alpha_I,\nu_I;\mathbf{x}) \qquad (2.2.6)
\end{aligned}$$

where the likelihood function is partitioned into three parts: $L_{SE}(\alpha,\beta,\kappa|\mathbf{x})$ (the contribution of the transitions from S to E), $L_{EI}(\alpha_E,\nu_E|\mathbf{x})$ (the contribution to the likelihood of the transitions from E to I), and $L_{IR}(\alpha_I,\nu_I|\mathbf{x})$ (the contribution to the likelihood function of transitions from I to R). Since the likelihood function produces very small values, we shall use the log-likelihood function:

$$\begin{aligned}
l(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) &= \log\left[L(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x})\right]\\
&= \log\left[L_{SE}(\alpha,\beta,\kappa|\mathbf{x})\right]+\log\left[L_{EI}(\alpha_E,\nu_E|\mathbf{x})\right] \qquad (2.2.7)\\
&\quad+\log\left[L_{IR}(\alpha_I,\nu_I|\mathbf{x})\right] \qquad (2.2.8)\\
&= l_{SE}(\alpha,\beta,\kappa;\mathbf{x})+l_{EI}(\alpha_E,\nu_E;\mathbf{x})+l_{IR}(\alpha_I,\nu_I;\mathbf{x}) \qquad (2.2.9)
\end{aligned}$$

The log-likelihood can be expressed as the sum of log-likelihood-parts, so to maximise the likelihood is to maximise the log-likelihood, which can be done by maximising each of the log-likelihood-parts. This greatly lowers the computation time for maximum-likelihood estimation, which is a method of model fitting if the full data are available (see next section), since this changes the problem from maximising a function in seven dimensions to two 2-dimensional maximisations and a 3-dimensional maximisation. In addition, the maximisation of each of the partial log-likelihoods can be performed in parallel on a parallel processor.

2.3 Model fitting methods

This thesis is concerned with the case where the transitions of hosts from the S state to the E state are unobserved, which is often the case in real world epidemics, for example, where there is a latent period (for example, [134, 145]). However, if the transitions to the E state are observed, then since the likelihood function would be tractable in this situation, maximum-likelihood estimation can be used to obtain parameter estimates. In this thesis, maximum-likelihood estimation will be embedded within our framework of model testing. This section details how maximum-likelihood estimation is performed within this thesis.

2.3.1 Maximum-Likelihood Estimation of Parameters for Complete Data

To maximise the log–likelihood, a common approach is to use a numerical optimisation algorithm. The Nelder-Mead Simplex algorithm [132] is one of the algorithms that have been used in this thesis to maximise the log-likelihood. As in the numerical literature, the optimisation algorithms presented here in this thesis are in the form used for function minimisation, but the same algorithm can be used for maximisation since maximisation of a function is the minimisation of the negative of the function. The Nelder-Mead Simplex algorithm is a gradient-free optimisation algorithms for multivariate functions. The algorithm involves the use of a simplex in n dimensions, which adapts itself at each iteration based on the function values at the points forming the simplex in order to find the local minimum. The algorithm consists of four operations: sorting, reflection, expansion, contraction. The general idea of the algorithm is that the simplex should extend on encountering a long slope, change direction on encountering a valley, and contract on encountering a minimum (using the description given in [109]). The algorithm was chosen because of its flexibility, and inclusion in many programming libraries (for example [98]).

The Nelder-Mead algorithm was selected over gradient-based alternatives such as Low Storage Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [114, 137], or other quasi-Newtonian algorithms, or the gradient-descent algorithm [42, 33] because a separate implementation of the gradient function is not needed, simplifying the computer implementation greatly. The L-BFGS (low storage BFGS) algorithm was also found to be effective at finding the maximum-likelihood estimate (and consistently found the same maximum as the Nelder-Mead algorithm). Gradient-based optimisation algorithms can be used in place of the non-gradient based methods used here, although there is a severe drawback in terms of maintaining and debugging the code for separate gradient calculations. Another alternative is to use the Nelder-Mead algorithm to obtain an estimate of the MLE and use a gradient based method to refine this estimate further, although the time saved in terms of computation may be outweighed by the labour involved in maintain and debugging a separate gradient function. Note that the maximisation of the likelihood is performed with bound constraints on the parameter values, which limits the choice of algorithms to those which can handle bound constraints.

Algorithm 1 (Nelder-Mead Simplex Algorithm for function minimisation)**.** *For a real valued function $f(\mathbf{x}) : x \in \mathbb{R}^n$, let $\rho > 0, \chi > \rho, 0 < \gamma < 1, 0 < \sigma < 1$ (note that in almost all implementations $\rho = 1, \chi = 2, \gamma = \frac{1}{2}, \sigma = \frac{1}{2}$). Then for each iteration k, perform the following steps:*

Sort *For each vertex in simplex Δ_k, order the vertices $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}\}$ such that*

$$f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \ldots \leq f(\mathbf{x}_{n+1})$$

Reflection *Let:*

$$\begin{aligned}\bar{\mathbf{x}} &= \frac{\sum_{i=1}^{n} x_i}{n} \\ \mathbf{x}_r &= \bar{\mathbf{x}} + \rho(\bar{\mathbf{x}} - \mathbf{x}_{n+1})\end{aligned}$$

Then, if $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$, $\mathbf{x}_{n+1} = \mathbf{x}_r$

Expansion *If* $f(\mathbf{x}_r) < f(\mathbf{x}_1)$*, let:*

$$\mathbf{x}_e = \bar{\mathbf{x}} + \chi(\mathbf{x}_r - \bar{\mathbf{x}}) = \bar{\mathbf{x}} + \chi\rho(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ *then* $\mathbf{x}_{n+1} = \mathbf{x}_e$ *else* $\mathbf{x}_{n+1} = \mathbf{x}_r$

Outside Contraction *If* $f(\mathbf{x}_n) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$*, let*

$$\mathbf{x}_c = \bar{\mathbf{x}} + \gamma(\mathbf{x}_r - \bar{\mathbf{x}}) = \bar{\mathbf{x}} + \gamma\rho(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

If $f(\mathbf{x}_c) < f(\mathbf{x}_r)$ *then* $\mathbf{x}_{n+1} = \mathbf{x}_c$ *and go to next iteration* $k + 1$*, else* $\mathbf{x}_{n+1} = \mathbf{x}_r$ *and go to* ***shrink*** *step*

Inside Contraction *If* $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$*, let*

$$\mathbf{x}_{cc} = \bar{\mathbf{x}} - \gamma(\bar{\mathbf{x}} - \mathbf{x}_{n+1})$$

If $f(\mathbf{x}_{cc}) < f(\mathbf{x}_r)$ *then* $\mathbf{x}_{n+1} = \mathbf{x}_{cc}$ *and go to next iteration* $k + 1$*, else* $\mathbf{x}_{n+1} = \mathbf{x}_r$ *and go to* ***shrink*** *step*

Shrink *Let*

$$\mathbf{x}_i = \mathbf{x}_1 + \sigma(\mathbf{x}_i - \mathbf{x}_1)$$

for all $i > 1$.

Note the log-likelihood function for this model is composed of the sum of separate independent parts (see equation 2.2.9) which only involve separate groups of parameters, so this likelihood can be maximised with respect to each of these groups separately, easing the computational burden greatly, as the computation time required to find the maximum-likelihood estimate increases sharply as the number of parameters increases.

The advantage of using the Nelder-Mead algorithm to find the maximum-likelihood estimates of the parameters in this thesis is that it does not require programming and maintenance of a separate gradient function, which is time-consuming particularly with the nature of the exploratory and experimental work performed for this thesis. Despite the fact that there is no satisfactory convergence theory for the Nelder-Mead algorithm, nor any guarantee that it will converge to a minimum [109, 122] (unless under very strict constraints), the Nelder-Mead algorithm has been used in a vast amount of research in many fields [105, 109, 21] - the original paper [132] has approximately 27,000 citations at the time of writing. Due to its popularity, the Nelder-Mead algorithm has been implemented in many numerical libraries [98, 149], and is included in the R and MATLAB language. In this thesis direct search methods are the only option for $l_{EI}(\alpha_E, \nu_E|\mathbf{x})$ and $l_{IR}(\alpha_I, \nu_I|\mathbf{x})$, as their gradient functions (which involve the derivative of the incomplete gamma function) are extremely difficult to implement in C++ in an accurate and efficient manner.

2.3.1.1 The Subplex Algorithm for Maximisation

The Nelder-Mead algorithm is an appropriate choice of algorithm for obtaining the MLE for a single data set. Unfortunately, when the maximisation needs to be performed for thousands of data sets, which is often the case when embedding maximum-likelihood estimation into another algorithm like MCMC, which will be used in this thesis for model testing, an algorithm is needed in which the maximum-likelihood estimator is found consistently for all the data sets without the need for human intervention. In some cases in this thesis, the Nelder-Mead is not sufficient for this purpose (for example, maximising the full likelihood within the model testing methods detailed later in this thesis). In these situations, the Subplex algorithm [155] for optimisation is a more appropriate choice of algorithm. The Subplex algorithm incorporates the Nelder-Mead algorithm within itself, and intends to improve convergence to the actual maximum based on two properties of the Nelder-Mead algorithm. First of all, the Nelder-Mead algorithm tends to be able to find the maximum with more ease in lower dimensional problems, than higher dimensional problems. Second of all, in certain cases of non-convergence, it has been found that restarting the Nelder-Mead algorithm at the last location, re-initialising the simplex, can allow the Nelder-Mead algorithm to converge to a maximum, dislodging the simplex from a situation where it has collapsed or got "stuck". The Subplex algorithm determines a step-size, and then divides the parameter space into several subspaces and maximises using the Nelder-Mead algorithm along each subspace. If a termination test (detailed later) is not satisfied, the step-sizes and subspaces are set again and the Nelder-Mead algorithm is restarted on these subspaces. This continues until the termination test is satisfied. A brief formal outline is as follows (as described in [155]):

Algorithm 2 (Subplex)**.** *In addition to the Nelder-Mead coefficients defined above (the default Nelder-Mead strategy used for Nelder-Mead algorithm embedded in the Subplex algorithm is* $\rho = 1, \chi = 2, \gamma = \frac{1}{2}, \sigma = \frac{1}{2}$*), let "*$scale$*" be a vector of step sizes. Let* ψ, ω *be the simplex reduction coefficients and step reduction coefficients respectively where* $0 < \psi < 1$ *and* $0 < \omega < 1$*. Let* $nsmin$ *and* $nsmax$ *be the minimum and maximum subspace dimensions. The default values of these settings (used in this thesis) are* $\psi = 0.25, \omega = 0.1, nsmin = \min(2, n), nsmax = \min(5, n)$*. Let* $\mathbf{x}$ *be the current approximation to the minimum.*

1. Determine step size (see Algorithm 3).
2. Set subspaces (see Algorithm 4).
3. Perform the Nelder-Mead algorithm on each subspace .
4. If

$$\frac{\max\left(\|\Delta\mathbf{x}\|_\infty, \|step \cdot \psi\|_\infty\right)}{\max(\|x\|_\infty, 1)} < tol$$

   where Δx is the difference between x and its value on the previous cycle of the algorithm, and tol is the error tolerance required, then the algorithm ends. Otherwise, go to step 1.

Algorithm 3 (Subplex (Setting the Stepsize))**.**

1. If the algorithm has just been started, that is, this is the first time that the step-size is being set, $step = scale$. Otherwise, perform the following steps:

(a) $$step \leftarrow \begin{cases} step \cdot \min\left(\max\left(\frac{\|\Delta\mathbf{x}\|_1}{\|step\|_1}, \omega\right), \frac{1}{\omega}\right) & nsubs > 1 \\ step \cdot \psi & nsubs = 1 \end{cases}$$

(b) for each component $step_i$ of $step$:

$$step_i \leftarrow \begin{cases} \text{sign}(\Delta x_i) \cdot |step_i| & \Delta x_i \neq 0 \\ -step_i & \Delta x_i = 0 \end{cases}$$

where $\|\mathbf{x}\|_1 = \sum_i |x_i|$.

Algorithm 4 (Subplex (Setting the subspaces))**.** *Let* $\Delta\mathbf{x}$ *be the vector of progress. Let* $nsmin$ *and* $nsmax$ *be the minimum and maximum subspace dimensions. Let there be* $subs$ *subspaces of* $ns_1, ns_2, \ldots ns_{subs}$ *dimensions (which sum to* n*), where* $\forall i \in \{1, 2, \ldots, subs\} : nsmin \leq ns_i \leq nsmax$*.*

1. Sort $\Delta\mathbf{x} = (\Delta x_1, \Delta x_2, \ldots, \Delta x_n)$ such that the largest component is first. Let this be denoted $\widetilde{\Delta\mathbf{x}} = (\widetilde{\Delta x}_1, \widetilde{\Delta x}_2, \ldots, \widetilde{\Delta x}_n)$.

2. Set
$$ns_1 = \underset{k \in K}{\arg\max} \left(\begin{cases} \frac{\|(\widetilde{\Delta x}_1, \ldots, \widetilde{\Delta x}_k)\|_1}{k} - \frac{\|(\widetilde{\Delta x}_{k+1}, \ldots, \widetilde{\Delta x}_n)\|_1}{n-k} & k \neq n \\ \frac{\|(\widetilde{\Delta x}_1, \ldots, \widetilde{\Delta x}_k)\|_1}{k} & k = n \end{cases} \right)$$
where $K = \{k | nsmin \leq k \leq nsmax$ and $nsmin \lceil (n-k)/nsmax \rceil \leq n-k\}$,$\|\mathbf{x}\|_1 = \sum_i |x_i|$. This step determines where the "gaps" are in $\widetilde{\Delta\mathbf{x}}$.

3. Repeat step 2 for $ns_2, ns_3, \ldots$ until $\sum_{i=1}^{subs} = n$.

An advantage of the simplex algorithm is that it allows bound constraints on values of $\mathbf{x}$. This was alluded to in the original description in [155], and implemented in the numerical C++ library NLOpt [98]. If any of the Nelder-Mead steps produces a simplex with a point outside of these constraints, the point is moved (usually to the nearest point within the constraints) such that the simplex lies within the constraints. This is useful as, in the likelihood we are maximising, $\alpha, \beta, \kappa > 0$. The motivation behind each of these steps in the Subplex algorithm lies far outside the scope of this thesis, and indeed form a chapter of the PhD thesis in which it was first described (for further details see [155]).

2.3.2 Maximum Likelihood Estimation and Model Comparison Methods: Challenges of Embedding Maximum Likelihood Estimation within Other Algorithms

The model comparison methods described later in this thesis involve what is essentially thousands of maximum-likelihood estimations, each on different data-sets. This produces additional challenges:

Since in this thesis, maximum-likelihood estimation is performed through numerical maximisation of the likelihood function, finding the global maximum of many functions in bulk is relatively difficult in comparison to maximising only one function. There are several reasons. First of all, there is absolutely no guarantee that the maximum found is a global maximum and not a local maximum. It is harder to check when maximum-likelihood estimation is performed on many data-sets in bulk because the only practical way to test convergence to the global maximum is to reset the maximiser and find the maximum-likelihood again starting at a different starting point. If the algorithm for numerical maximisation outputs a different minimum (recall that numerical maximisation of the likelihood function performed by using a numerical minimisation algorithm on the negative of the log likelihood function) and this minimum is even more negative, it would mean that the maximum likelihood estimate had not been reached in the previous performance of the algorithm. Performing the numerical maximisation several times in order to verify the global maximum increases and multiplies the computation time needed to get an accurate result. Choosing such an algorithm for finding the maximum likelihood estimate for thousands of different data-sets is not an easy task. This is because in addition to verification of the global maximum being totally automated, the choice of starting point for the numerical algorithm needs to be totally automated. Within numerical optimisation literature there are very few global maximisers (a term specific to numerical optimisation and not to be confused from normal usage) which means that regardless of which starting point the optimiser is started at, the algorithm will converge to a local maximum. There is also the problem of finding an optimiser which does so in a reasonable number of iterations. For example, the simulated annealing algorithm will find a maximum, but may take many thousands of iterations to converge. When performing maximum-likelihood estimation in bulk, there is the problem that each likelihood is different in terms of mathematical properties. One log-likelihood function may have the properties necessary to allow the optimiser to converge, but another may cause optimiser to never converge to a stable value. This is a problem with certain optimisers such as the Newton type algorithms, for example, which need the function to be approximately locally quadratic.

In order to combat these problems we use the following algorithm for each maximum likelihood estimation (performed in bulk) within the algorithms (for model comparison) in this thesis:

1. Utilise the optimisation algorithm (Subplex) to obtain an estimate of the MLE (within given stopping criteria and tolerances)

2. Verify that this is the maximum likelihood estimate by restarting the optimiser at a different starting point. If the optimiser produces a point with the same likelihood value (subject to tolerances that have been prespecified) accept the current estimate as the maximum likelihood estimate. Otherwise, take the point which produces the largest likelihood value and repeat this process again from the start.

This provides some verification that the estimate produced by the optimiser is the maximum likelihood estimate. This algorithm appears to eliminate or minimise most of the problems mentioned above. In most cases, only two optimisations were needed to obtain two matching values. However, sometimes the optimiser would perform successive optimisations for over half an hour to get two successive matching values.

In this thesis we used the Subplex algorithm, which is based on the Nelder-Mead algorithm, which is a heuristic method. From the experience of many practitioners, in practice the Subplex algorithm is able to find the maximum of a function from a wide variety of starting points and find the maximum in functions which are difficult to optimise. However, few mathematical results have been derived that prove that the optimiser will always reach a maximum regardless of starting point, or results which show the conditions under which a maximum can be reached. At present such results exist in literature involve showing that the Nelder-Mead algorithm will do so in simple cases, but in complicated cases as found here it is much more difficult to analytically derive any sort of results for this algorithm. This algorithm, was tested on some runs against a gradient based algorithm and does converge to the maximum value that is found by gradient-based methods in almost every case, and does not require the programming and maintenance for code for a derivative function which makes the probability of coding errors a lot less. Measures have been taken to mitigate these drawbacks for example the restarting the simplex from many points to verify that the maximum found is a global maximum. The Nelder-Mead algorithm can run for many thousands of iterations with no improvement in function value, giving the illusion of convergence. The risk is mitigated by restarting the simplex every time the optimiser appears to have converged. This has been incorporated into the Subplex algorithm.

2.3.3 Bayesian Inference Using Data Augmentation

Incomplete data are commonly encountered within epidemic modelling. Often, the exact transition times are not observed (for example: [36, 134, 143, 145, 172]). This can arise because of the nature of the infection: an example is in the SEIR model, where the transition from the susceptible state S to the asymptomatic infection state E cannot be observed. Depending on the observation process, some transitions between states may not be observable. For example, in the situation of a disease with a long latent period in which symptoms are not visible, only the transitions from the E state to the I state are observed. A similar situation which the transition times are not exactly known arises when there is a delay in diagnosis or reporting of the disease. This can be the case especially when the symptoms of the disease in its early stages resembles many other diseases (for example, foot-and-mouth disease, in [100, 96, 97]). Another situation in which the transition times cannot be precisely recorded is that of snapshot data. In this type of data the transition time is not observed directly; instead the disease status of individual hosts is recorded at fixed sampling times, providing what is known as a "snapshot" of the epidemic at these times. The host is known to not have made the transition before a certain snapshot and is only known to have made the transition between two consecutive snapshots. For examples of epidemics where this has been the case see [73, 134, 53, 145]. Combination of these missing data types may occur, for example in the SEIR epidemic model, where transition times to the I state may not be observed directly, but only constrained to lie between successive sampling times. Another type of unobserved data occurs in which the data only records the times at which individuals are removed from the population (for example [64, 141, 143]). This is known as "removal-only data". This situation occurs frequently in infections which are quarantined or culled when an infection is detected, but there is a delay between the infection time and the time that the infection is detected. The final size of the epidemic is often unknown. If the disease has a latent period, or there is a delay in reporting or diagnosis, there will be cryptic infections - infected hosts which are infected but are not detected or asymptomatic.

In this thesis, in which spatio-temporal epidemics are the focus, these types of missing data are very common. Because of this, Bayesian statistics and data augmented MCMC is often used to fit models. In this chapter, a brief outline of data augmented MCMC will be given. In later chapters, the model assessment methods developed in this thesis will be developed to fit into this framework.

For the past 20 years, model fitting for incomplete data has predominantly been performed using Bayesian methods. In Bayesian inference, all quantities including model parameters are treated as random variables with distributions which are updated in the light of observation, using the laws of conditional probability. The distribution assigned at any time to a parameter represents belief about its value in the light of all available information on the parameter up to that time.

This interpretation of probability leads to a different approach in model fitting. In Bayesian statistics, rather than trying to estimate the true value of a parameter, the aim is to obtain a distribution which represents the belief about the value of the unknown parameter. To do so requires specification of a distribution which represents the initial belief about the value of the unknown parameter, known as the *prior distribution*. This prior distribution is combined with the information in the data through Bayes theorem to obtain what is known as a *posterior distribution*, which represents the belief of the parameter's value given the data. More formally: let $\boldsymbol{\theta}$ represent the parameter vector of interest and let $\boldsymbol{y}$ represent the observed data. According to Bayes theorem, the posterior distribution $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ is given by,

$$\begin{aligned} \pi(\boldsymbol{\theta}|\boldsymbol{y}) &= \frac{\pi(\boldsymbol{y}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta})}{f(\boldsymbol{y})} \\ &\propto \pi(\boldsymbol{y}|\boldsymbol{\theta}) \cdot \pi(\boldsymbol{\theta}) \end{aligned}$$

where

$$f(\boldsymbol{y}) = \int_{\Theta} \pi(\boldsymbol{y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta}$$

The posterior distribution is often only calculated up to a constant of proportionality; since it is a probability distribution, it must integrate to unity. In addition, the denominator of the above formula may be difficult to calculate. For more details about Bayesian statistics, refer to [58, 19].

However, in epidemic modelling, there is often missing data. In this case, data augmentation (originally developed for use with the EM algorithm but later extended for use with MCMC and RJMCMC [44, 178, 56, 179]) can be used (for example [65, 64, 143, 172, 52, 66, 166, 36, 134, 32, 96, 97, 145, 95, 100, 104, 107, 63]). Following this approach, the missing data are taken as extra parameters $\boldsymbol{z}$, with the "complete" data $\boldsymbol{x}$ being comprised of the observed data $\boldsymbol{y}$ and unobserved data $\boldsymbol{z}$:

$$\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$$

The posterior density $\pi(\boldsymbol{\theta}|\boldsymbol{y})$ can be obtained by integrating out the missing data [65, 143, 64, 27, 172, 171]:

$$\begin{aligned}\pi(\boldsymbol{\theta}|\boldsymbol{y}) &\propto \pi(\boldsymbol{\theta})\int \pi(\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,d\boldsymbol{z}\\ &= \pi(\boldsymbol{\theta})\int \pi(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{\theta})\pi(\boldsymbol{z}|\boldsymbol{\theta})\,d\boldsymbol{z}\end{aligned}$$

The integral $\int \pi(\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,d\boldsymbol{z}$ in the above equation may have no analytical solution. Nevertheless, the joint density $\pi(\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\theta})$ may be analytically tractable. Therefore, we can construct iterative methods for sampling from this joint distribution, obtaining information about $\pi(\theta|\boldsymbol{y})$ from the samples obtained, and thus obtain estimates of posterior summary statistics such as mean or variance of a parameter, or histogram estimates of univariate or multivariate marginal distributions.

The Monte Carlo method [79, 159] is used to obtain estimates of such summary statistics. Monte Carlo is the technique of generating a large number of samples from the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{x})$, and calculating an estimator of the statistic of interest. For example, the Monte Carlo estimator for the posterior mean of $g(\boldsymbol{\theta})$ (where $\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_m$ are samples from the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{x})$) is given by:

$$\mathbb{E}_\pi(g(\boldsymbol{\theta})) \approx \frac{1}{m}\sum_{i=1}^{m} g(\boldsymbol{\theta}_i)$$

The law of large numbers makes this estimator converge to the actual values. This method is straightforward if it is possible to sample from the posterior distribution directly. However, it may not be possible to sample directly from the distribution of interest. For this situation, Markov chain Monte Carlo (MCMC) is used.

MCMC is the method whereby one formulates a Markov chain such that its stationary distribution is the distribution of interest (which in epidemic modelling is the posterior distribution $\pi(\boldsymbol{\theta}|\mathbf{x})$). Readers may recall that a Markov chain is a stochastic process which is “memoryless”. That is, more formally (see [77] for further information on the theory behind MCMC):

Definition 5 (Markov Chain)**.** A *Markov chain i*s a sequence of random variables $\{X_t\}$, where $t = 0, 1, 2, \ldots$ such that

$$\Pr(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_0 = x_0) = \Pr(X_t = x_t | X_{t-1} = x_{t-1})$$

In other words the current state of a Markov Chain is only dependent on its previous state. If the Markov chain is irreducible and aperiodic, the Markov chain will converge in distribution to the stationary distribution.

Definition 6 (Irreducible)**.** A Markov chain is *irreducible* if $\forall x_1, x_2\ \exists t < \infty$: $\Pr(X_t = x_2 | X_0 = x_1) > 0$.

Definition 7 (Period)**.** The *period* of state x, denoted period(x) is defined as:

$$\text{period(x)} = \gcd(\{t \geq 1 | \Pr(X_t = x | X_0 = x) > 0\})$$

Definition 8 (Aperiodic)**.** A Markov Chain is *aperiodic* if all elements have period 1.

It is assumed that these conditions are always met. There are many algorithms that can create an aperiodic irreducible Markov chain that has a stationary distribution equal to a desired distribution, the most commonly used algorithm is the Metropolis Hastings algorithm [127, 85] (for a good overview of its usage within Bayesian statistics see [154]). The Metropolis-Hastings algorithm generates an aperiodic irreducible Markov chain with a stationary distribution equal to any desired target density, as long as the target density is known up to a constant of proportionality.

However, the Metropolis-Hastings algorithm is not totally suitable for fitting spatio-temporal epidemic data in which the infection events are unobserved. Since the final size of the epidemic is unknown, the unobserved component of the augmented data z, now treated as a nuisance parameter, is a vector of unknown size. The Metropolis-Hastings algorithm is not designed for parameter vectors of unknown size. An extension of the Metropolis-Hastings algorithm known as Reversible jump MCMC [75] can be used in this situation (for example [65, 64, 143, 172, 52, 66, 166, 36, 134, 32, 96, 97, 145, 95, 100, 104, 107, 63]).

The original paper [75], explained that a model with a parameter space of variable dimension can be considered as a hierarchical model consisting of sub-models which have parameter vectors $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3 \ldots,$ of different dimension. Thus, an epidemic model with unknown cryptic infections can be thought of as a hierarchical model containing sub-models of dimension $k = 1, \ldots, k_{max}$ where $k_{max}$is the maximum possible number of cryptic infections. The parameters of each sub-model $\boldsymbol{\phi}_k = (\boldsymbol{\theta}, \mathbf{z}_k)$, where the z_k is the vector of unobserved asymptomatic infection times corresponding to the case where the epidemic has k cryptic infections. In [75], the following is used to generate a Markov chain for the posterior distribution using the following reasoning.

Given the current distribution of $\boldsymbol{\phi}'$,$P^{(t)}(\boldsymbol{\phi}')$, at time t of the Markov chain, the state of the chain at time $t+1$ is the density $P^{(t+1)}(\boldsymbol{\phi}') = \int T(\boldsymbol{\phi}'|\boldsymbol{\phi})P^{(t)}(\boldsymbol{\phi})d\boldsymbol{\phi}$ ($T(\boldsymbol{\phi}'|\boldsymbol{\phi})$ is the transition kernel of the Markov chain), and the stationary distribution π satisfies $\pi(\boldsymbol{\phi}') = \int T(\boldsymbol{\phi}'|\boldsymbol{\phi})\pi(\boldsymbol{\phi})d\boldsymbol{\phi}$. This means transitioning from one time step of the Markov chain does not affect the stationary distribution. Consider only reversible Markov chains, where the detailed balance equation must hold:

$$T(\boldsymbol{\phi}|\boldsymbol{\phi}')\pi(\boldsymbol{\phi}')d\boldsymbol{\phi}' = T(\boldsymbol{\phi}'|\boldsymbol{\phi})\pi(\boldsymbol{\phi})d\boldsymbol{\phi}.$$

Similar to the Metropolis-Hastings algorithm, the move to the next sub-model, which is proposed from distribution $j(k'|k)$, the next sub-model having k' exposure (asymptomatic infection) events and the current sub-model having k events, consists of proposing $\boldsymbol{u}$ and $\boldsymbol{u}'$, proposal vectors of length r and r' such that $k + r = k' + r^{'}$, from its known proposal distribution $g_{k \to k'}(\boldsymbol{u})$. It is then combined with current state $\boldsymbol{\phi}$ with some known deterministic function $h(\boldsymbol{\phi}, \boldsymbol{u})$ to give a proposed new state $(\boldsymbol{\phi}', \boldsymbol{u}')$. This proposed new state is accepted with probability $\alpha\left((\boldsymbol{\phi}^{'}, \boldsymbol{u}')|(\boldsymbol{\phi}, \boldsymbol{u})\right)$. To make this algorithm converge to the desired stationary distribution, the following must be satisfied:

$$\begin{aligned} j(k|k')g_{k'\to k}(\boldsymbol{u})\alpha\left((\boldsymbol{\phi},\boldsymbol{u})|(\boldsymbol{\phi}^{'},\boldsymbol{u}')\right)\pi(\boldsymbol{\phi}')\,d\boldsymbol{u}'\,d\boldsymbol{\phi}' \\ = j(k'|k)g_{k\to k'}(\boldsymbol{u}')\alpha\left((\boldsymbol{\phi}^{'},\boldsymbol{u}')|(\boldsymbol{\phi},\boldsymbol{u})\right)\pi(\boldsymbol{\phi})\,d\boldsymbol{u}\,d\boldsymbol{\phi} \end{aligned}$$

Applying the change of variable rule to the left-hand side we obtain:

$$\begin{aligned} j(k|k')g_{k'\to k}(\boldsymbol{u})\alpha\left((\boldsymbol{\phi},\boldsymbol{u})|(\boldsymbol{\phi}^{'},\boldsymbol{u}')\right)\pi(\boldsymbol{\phi}')\left|\frac{\partial h(\boldsymbol{\phi}^{'},\boldsymbol{u}')}{\partial(\boldsymbol{\phi},\boldsymbol{u})}\right|\,d\boldsymbol{u}\,d\boldsymbol{\phi} \\ = j(k'|k)g_{k\to k'}(\boldsymbol{u}')\alpha\left((\boldsymbol{\phi}^{'},\boldsymbol{u}')|(\boldsymbol{\phi},\boldsymbol{u})\right)\pi(\boldsymbol{\phi})\,d\boldsymbol{u}\,d\boldsymbol{\phi} \end{aligned}$$

which is only satisfied if

$$\begin{aligned} j(k|k')g_{k\to k'}(\boldsymbol{u}')\alpha\left((\boldsymbol{\phi}^{'},\boldsymbol{u}')|(\boldsymbol{\phi},\boldsymbol{u})\right)\pi(\boldsymbol{\phi}) = j(k'|k)g_{k'\to k}(\boldsymbol{u})\alpha(\boldsymbol{\phi}',\boldsymbol{\phi})\pi(\boldsymbol{\phi}')\left|\frac{\partial h(\boldsymbol{\phi}^{'},\boldsymbol{u}')}{\partial(\boldsymbol{\phi},\boldsymbol{u})}\right| \\ \Rightarrow \frac{\alpha\left((\boldsymbol{\phi}^{'},\boldsymbol{u}')|(\boldsymbol{\phi},\boldsymbol{u})\right)}{\alpha\left((\boldsymbol{\phi},\boldsymbol{u})|(\boldsymbol{\phi}^{'},\boldsymbol{u}')\right)} = \frac{j(k'|k)g_{k'\to k}(\boldsymbol{u})\pi(\boldsymbol{\phi}')}{j(k|k')g_{k\to k'}(\boldsymbol{u}')\pi(\boldsymbol{\phi})}\left|\frac{\partial h(\boldsymbol{\phi}^{'},\boldsymbol{u}')}{\partial(\boldsymbol{\phi},\boldsymbol{u})}\right| \end{aligned}$$

Therefore, [75] selects $\alpha\left((\boldsymbol{\phi}^{'},\boldsymbol{u}')|(\boldsymbol{\phi},\boldsymbol{u})\right) = \min\left\{1, \frac{j(k'|k)g_{k'\to k}(\boldsymbol{u})\pi(\boldsymbol{\phi}')}{j(k|k')g_{k\to k'}(\boldsymbol{u}')\pi(\boldsymbol{\phi})}\left|\frac{\partial h(\boldsymbol{\phi}^{'},\boldsymbol{u}')}{\partial(\boldsymbol{\phi},\boldsymbol{u})}\right|\right\}$

Thus, to generate a Markov Chain which converges to the stationary distribution corresponding to the posterior distribution $\pi(\boldsymbol{\theta},\boldsymbol{z}|\boldsymbol{y})$ the following algorithm can be used:

Let $\boldsymbol{\phi} = (\boldsymbol{\theta},\boldsymbol{z})$. The Reversible Jump MCMC algorithm requires as input k and $\boldsymbol{\phi}_0$ as starting values. Choose proposal distributions j and g.

1. Propose the move to $k^{'}$ cryptic infections from k cryptic infections, from proposal distribution $j(k'|k)$.

2. Let $\boldsymbol{u}$ and $\boldsymbol{u}'$ be vectors of length r and r' such that $k + r = k' + r^{'}$. Sample $\boldsymbol{u}$ from proposal distribution with joint density $g_{k\to k'}(\boldsymbol{u})$ (the probability density of the reverse move is $g^{'}_{k\to k'}(\boldsymbol{u}^{'})$).

3. Generate a proposed $\phi^{'}_{k'}$ from the diffeomorphism $(\boldsymbol{\phi}^{'}_{k'}, \boldsymbol{u}') = h(\boldsymbol{\phi}_k, \boldsymbol{u})$.

4. Accept the move $(\boldsymbol{\phi}^{'}_{k'}, \boldsymbol{u}')$ with probability

$$\alpha\left((\boldsymbol{\phi}^{'}_{k'}, \boldsymbol{u}')|(\boldsymbol{\phi}_k, \boldsymbol{u})\right) = \min\left\{1, \frac{\pi(k^{'}, \boldsymbol{\phi}^{'}_{k'}|\boldsymbol{x})}{\pi(k, \boldsymbol{\phi}_k|\boldsymbol{x})} \frac{j(k|k^{'})}{j(k'|k)} \frac{g^{'}_{k\to k'}(\boldsymbol{u}^{'})}{g_{k\to k'}(\boldsymbol{u})} \left|\frac{\partial(\boldsymbol{\phi}^{'}_{k'}, \boldsymbol{u}')}{\partial(\boldsymbol{\phi}_k, \boldsymbol{u})}\right|\right\}$$

For the SEIR model example used in this chapter, the above algorithm is simplified greatly by making it only possible at each state in the Markov chain only to add a cryptic infection or delete a cryptic infection, (moves in which the number of cryptic infections is kept the same reduce to a standard Metropolis-Hastings update step). Therefore, $\boldsymbol{u}$ will always consist of at most a single random number. To simplify things further g is chosen as the uniform distribution and h simply maps $\boldsymbol{u}$ onto the added infection time without any transformation. Thus, the Jacobian term, acceptance probability, and the algorithm are greatly simplified. The method detailed below is based upon the work [64, 143] adapted to an SEIR spatial epidemic model. The algorithm can be easily adapted to work with SIR, SI, and any other type of compartmental model.

For each iteration, the following process is repeated:

1. Let the current state of the algorithm at time $k = 0, 1, 2, \ldots$ be denoted by $(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})$ where $\boldsymbol{\theta}^{(k)} = (\alpha^{(k)}, \beta^{(k)}, \kappa^{(k)}, \mu_E^{(k)}, \sigma_E^{(k)}, \mu_I^{(k)}, \sigma_I^{(k)})$ and the times of transition into the exposed state be denote by $\mathbf{z}^{(k)} = z_1^{(k)}, z_2^{(k)}, \ldots, z_N^{(k)}$ where N is the number of hosts where N is fixed (and hence the length of $\boldsymbol{z}^{(k)}$ is fixed).

2. Using the Metropolis-Hastings algorithm, update each of the parameters in the parameter vector individually. That is, for each parameter $\phi^{(k)}$ in $\boldsymbol{\theta}^{(k)}$:

   (a) Draw a proposal value ϕ' from the proposal distribution $q(\phi'|\phi^{(k)})$.

   (b) Calculate the probability of acceptance α. Let $\boldsymbol{\theta}^*$ be $\boldsymbol{\theta}_t$ with $\phi^{(k)}$ replaced by ϕ':

$$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^*|\mathbf{z}^{(k)}, \mathbf{y}) \cdot q(\phi'|\phi^{(k)})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y}) \cdot q(\phi^{(k)}|\phi')}\right)$$

```
    (c) With probability $\alpha$, set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^*$, otherwise set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)}$.

3. For each data item $z_i^{(k)}$ in $\mathbf{z}^{(k)}$:

    (a) **if** $t_I^{(i)} < T$ set move type to *Standard*.
    (b) **else :**
        i. **if** $z_i^{(k)}$ does not fall within $[0, T]$ (which is only the case when the
           transition does not happen within $[0, T]$ and thus $z_i^{(k)}$ is set to $\infty$)
           set move type to *Addition*
        ii. **else** set move type to *Shift* or *Deletion* with probability $\frac{1}{2}$
    (c) **if** move type is *Standard, Addition* or *Shift*: Generate proposal $z_i^{(k)*} \sim$
        $\mathrm{Unif}(0, T)$
    (d) **else** set proposed value of $z_i^{(k)*}$to be outside $[0, T]$
    (e) **if** move type is *Standard* or *Shift* set $\nu = 1$
    (f) **else if** move type is *Addition* $\nu = \frac{T}{2}$
    (g) **else if** move type is *Deletion* set $\nu = \frac{2}{T}$
    (h) Set $\mathbf{z}^* = z_1^{(k)}, \ldots, z_i^{(k)*}, \ldots, z_N^{(k)}$. Then the acceptance probability $\alpha$ is
        given by:

        $$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^*, \mathbf{y})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y})} \cdot \nu\right)$$

    (i) With probability $\alpha$, set $\mathbf{z}^{(k+1)} = \mathbf{z}^{*c}$ otherwise $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}$
```

The above algorithm can be modified to make it more efficient, for example by using an independence sampler for the proposal distributions for the transition times to the infectious state [140, 107]:

$$q(I_i - z_i, I_i - z_i^*) \equiv \mathrm{Gamma}(\alpha_I, \nu_I)$$

$$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^*, \mathbf{y})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y})} \cdot \frac{q(\mathbf{z}^*|\mathbf{z}^{(k)})}{q(\mathbf{z}^{(k)}|\mathbf{z}^*)}\right)$$

for the cases where it is known that $I_i < T$.

In practice, whilst it may reduce the amount of autocorrelation between the MCMC samples, it would be computationally expensive to update all of the augmented data items per cycle of the algorithm. In practice, only a percentage of the exposure times are updated per cycle of the algorithm, usually chosen randomly. The percentage of the augmented data which is updated per cycle of the algorithm is chosen such that there is a balance in the trade-off between computation time and chain mixing. There appear to be no results giving a theoretical level of augmented data which should be updated per cycle, so this percentage is usually determined approximately by running several runs and determining what the trade-off is in terms of computation time versus chain mixing [107].

It should also be noted that there will be posterior correlations between the various parameters in the model. For example there will be a correlation in the posterior distribution between the kernel parameter and the secondary infection rate. In addition, there will be dependence between the imputed infection times and the parameters which determine infectious challenge. As a result, the chains produced by RJMCMC and data augmentation methods usually have high autocorrelation or poor mixing. Work has been done on this topic to reduce the amount of correlations between parameters, but is outside the scope of this thesis. In the context of this thesis, the algorithms presented here provide sufficiently low autocorrelation and low posterior correlations between model parameters for our purposes, which is to assess the effectiveness of model comparison and model assessment methods. For more information, see work done on partially non-centred parametrisations [107, 97]. The methods presented here only requires that samples are generated from $\pi(\theta, \boldsymbol{z}|\boldsymbol{y})$ - how the samples are generated is unimportant.

There has been much work on efficient MCMC for stochastic epidemic models, but this algorithm is favoured because of its simplicity and adequate mixing on models involved. More complex algorithms can be used [107, 97, 108, 130] but

Parameter	Actual values
α	0.001
β	3.000
θ	0.030
μ_E	5.000
σ_E^2	2.500
μ_I	1.772
σ_I^2	0.858

Table 2.1: Parameter values used to generate the simulated data. Parameters on the left column with the actual values used to generate the data in the right-hand column.

there is a trade-off between the advantages and the disadvantages in using them.

2.4 Example: Comparison of Maximum-Likelihood and Full Bayesian Approach

To illustrate the estimation process, a data-set has been generated using the spatio-temporal Gillespie algorithm above, with the parameters given in Table 2.1. A plot of the data can be found in 2.4.1.

Maximum-Likelihood Estimation can be used on the full data, if the exposure times are known, which in this case is possible because the data has been generated artificially through an algorithm. In a real-world situation it is difficult to get observations of the exposure times if the infection is asymptomatic, unless for example, if it is possible to test to see if the pathogen is present directly.

For the demonstration of maximum-likelihood estimation given here all the transition times are "observed". However, for the demonstration of RJMCMC given here, the observations consist of the times of entry into the infectious I and removed R states, but the times of the transitions into the exposed state E are not known, which corresponds to a situation where tests that detect symptoms perfectly are applied with arbitrarily high frequency. In this example, the data was observed until all hosts entered the removed state. The RJMCMC algorithm detailed earlier can be applied to snapshot data as well, with a few modifications. Posterior means and variances are available for the unobserved exposure times, but it would be impractical to list these results here. The reversible jump MCMC

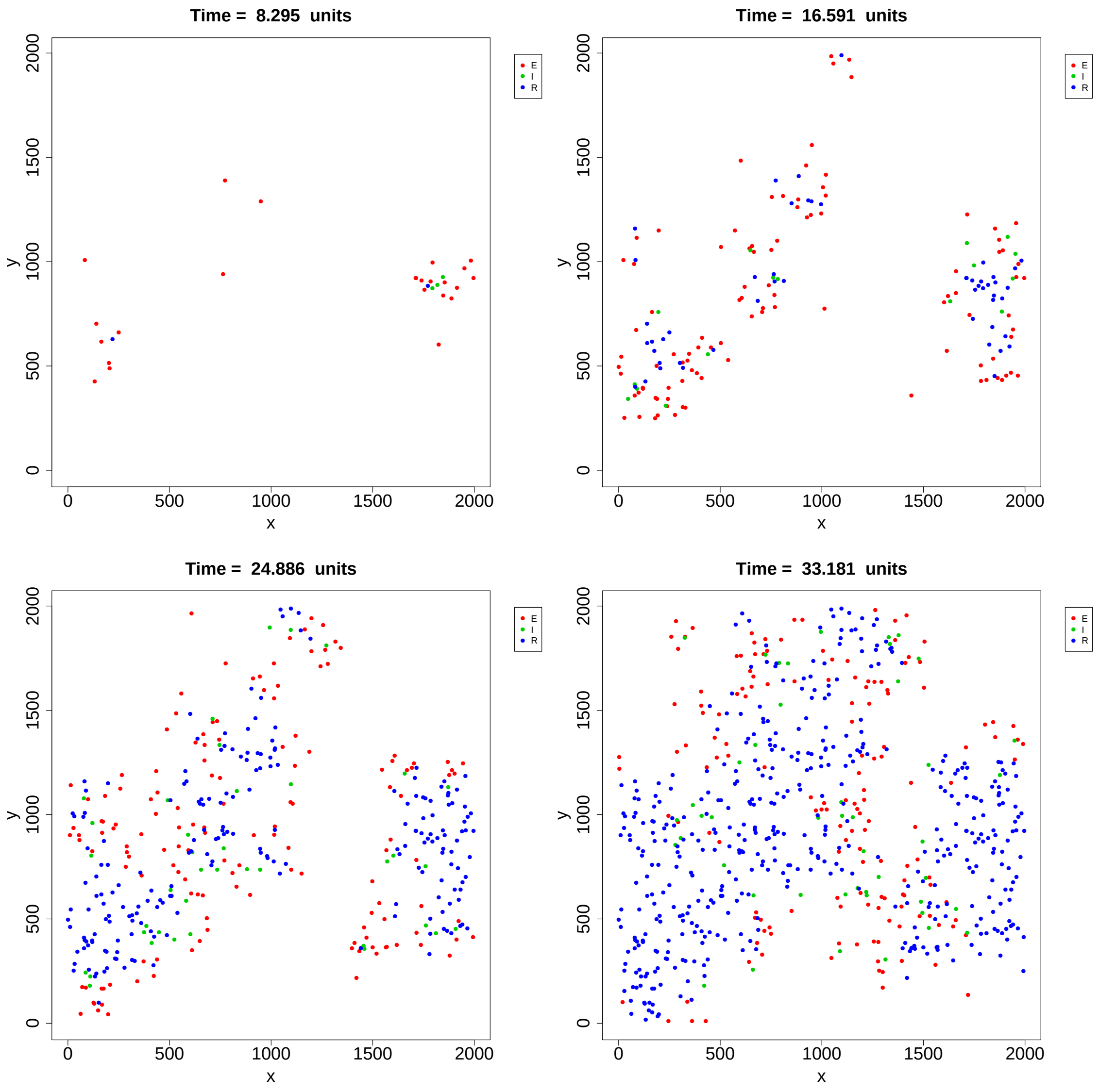

Figure 2.4.1: Snapshots of the simulated epidemic generated with an exponential kernel. Each point on the graph represents one host. Points are colour-coded to represent the current state of the host. Susceptible points are not displayed to maintain clarity of the graph. The colour of the points on the graph indicate the state of each host at the given time. Red indicates the host is exposed, green indicates the host is infectious and blue indicates that the host is removed.

algorithm used is the algorithm detailed above in the previous section.

The results of the maximum-likelihood estimation (using simulated data in which all the transition times are observed) are as shown in Table 2.2. The Nelder-Mead algorithm was used to obtain maximum-likelihood estimates but note that a gradient-based method would be also suitable for the maximum-likelihood estimation. The maximum-likelihood estimation took approximately one second. As a check that the results from the maximum-likelihood estimation algorithm were reasonable, which in turn provides support that the algorithm was correctly coded, the deviance was calculated and the resulting p-value was calculated, as this is known to come approximately from the chi-square distribution (Likelihood Ratio test with Wilks' theorem):

$$-2\log(\Lambda) \sim \chi^2_7$$

where

$$\Lambda = \frac{L(\mathbf{x}|\boldsymbol{\theta}_{act})}{L(\mathbf{x}|\hat{\boldsymbol{\theta}})}$$

and $L(\mathbf{x}|\boldsymbol{\theta}_{act})$ is the likelihood evaluated at the actual known parameter values and $L(\mathbf{x}|\hat{\boldsymbol{\theta}})$ as the likelihood evaluated at the maximum-likelihood estimate obtained by running the computer code. The value of the deviance was 3.079372 with a p-value of 0.877564 which shows that the estimates of the maximum-likelihood estimator through the Nelder-Mead algorithm appear to be not significantly different from the actual parameter values and thus showing that the computer code is giving results which seem reasonable. This shows that the true parameter vector would lie in a confidence interval of any reasonable confidence level. This gives support that the algorithm is correctly coded, although it is not absolute proof.

The reversible jump MCMC algorithm was run for 10,678,487 iterations which took approximately 4 hours. Details on methods used to accelerate the reversible jump MCMC algorithm will be detailed in future chapters, but readers must

Parameter	Actual values	MLE
α	0.001	0.001
β	3.000	3.012
θ	0.030	0.030
μ_E	5.000	5.033
σ_E^2	2.500	2.437
μ_I	1.772	1.798
σ_I^2	0.858	0.907

Table 2.2: Table of maximum-likelihood estimates obtained from the full data.

Parameter	Actual values	Posterior Mean	Posterior SD
α	0.001	0.000827	0.000233
β	3.000	3.4995873	0.5120371
θ	0.030	0.0311221	0.0012444
μ_E	5.000	5.0058293	0.1224685
σ_E^2	2.500	2.9744470	0.3519408
μ_I	1.772	1.8066625	0.0326887
σ_I^2	0.858	0.9252789	0.0546392

Table 2.3: Table of posterior means and variances obtained from the data with exposures unobserved.

note that an unoptimised implementation would take 200 times the runtime and a optimised parallel implementation would take 16 times the runtime. After discarding a conservative burn-in of 1 million updates, and thinning the output by a factor of 7, the posterior mean and the standard deviation estimates can be found in table 2.3. Readers should observe that the true parameter values fall within two posterior standard deviations of the posterior mean. Posterior densities and trace plots of the MCMC can be found in fig. 2.4.2 and fig. 2.4.3. Bivariate plots showing the posterior correlations between model parameters can be found in fig. 2.4.4. As expected, the parameters β and κ display some correlation, which can be reduced through normalisation of the transmission kernel, at the cost of making the results more difficult to interpret. Readers should note that one of the advantages of using data-augmented reversible-jump MCMC is that one can obtain samples from the posterior distribution of the unobserved data. This is crucial for the work presented later in this thesis in which classical tests of model fit are embedded in this framework.

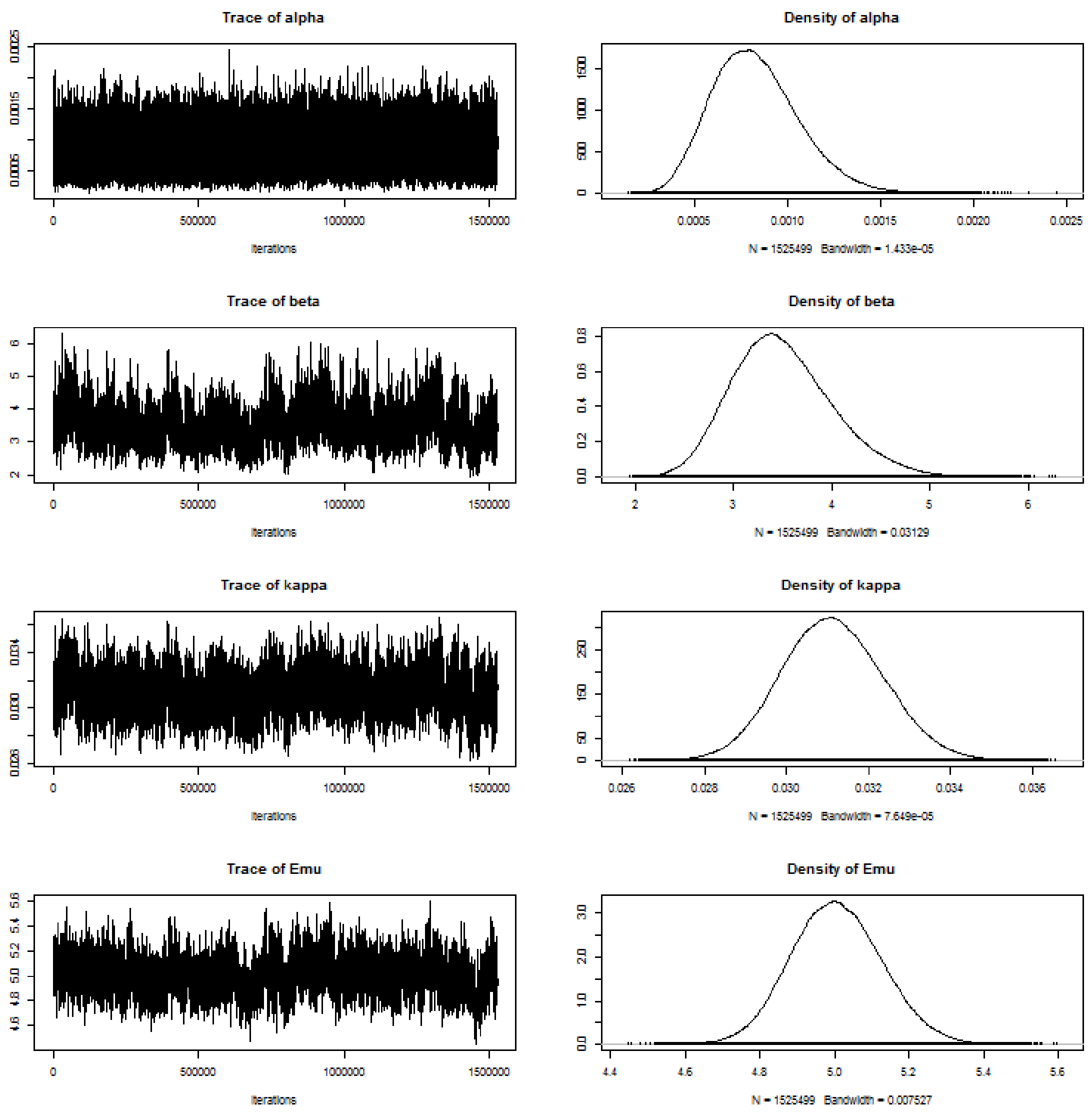

Figure 2.4.2: Trace and Kernel density plots for the posterior distributions of the model parameters (1 of 2 plots).

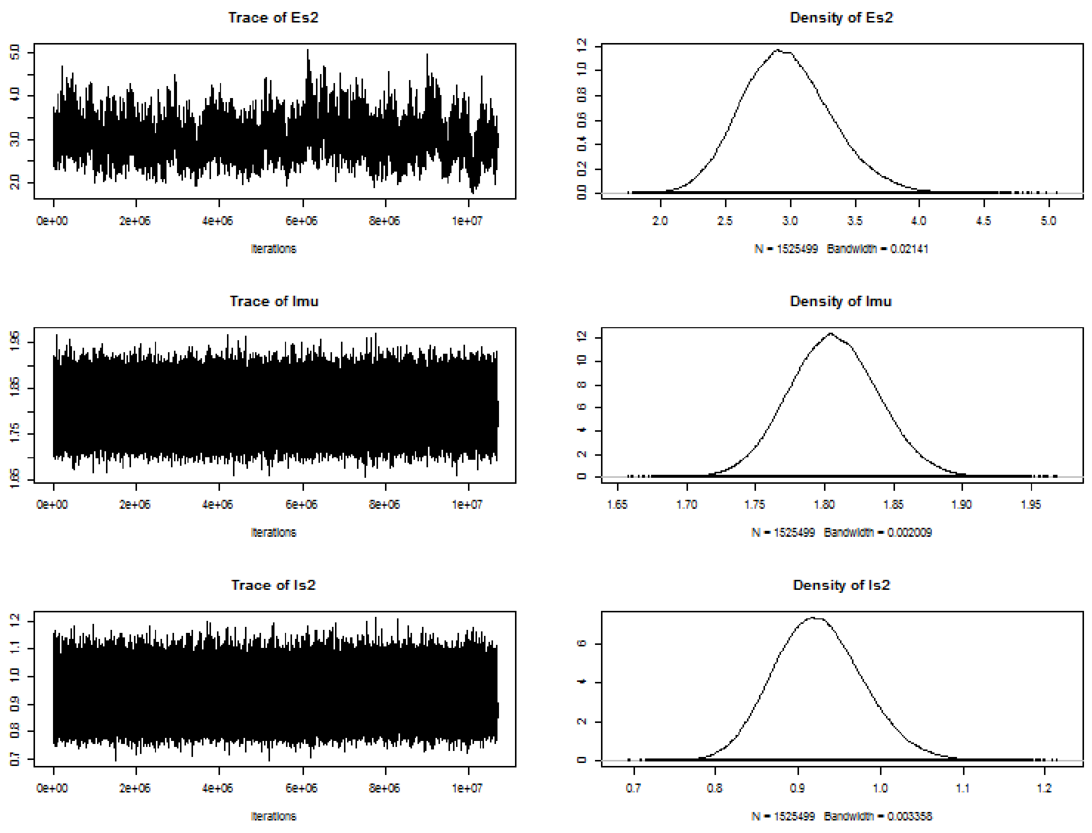

Figure 2.4.3: Trace and Kernel density plots for the posterior distributions of the model parameters (2 of 2 plots).

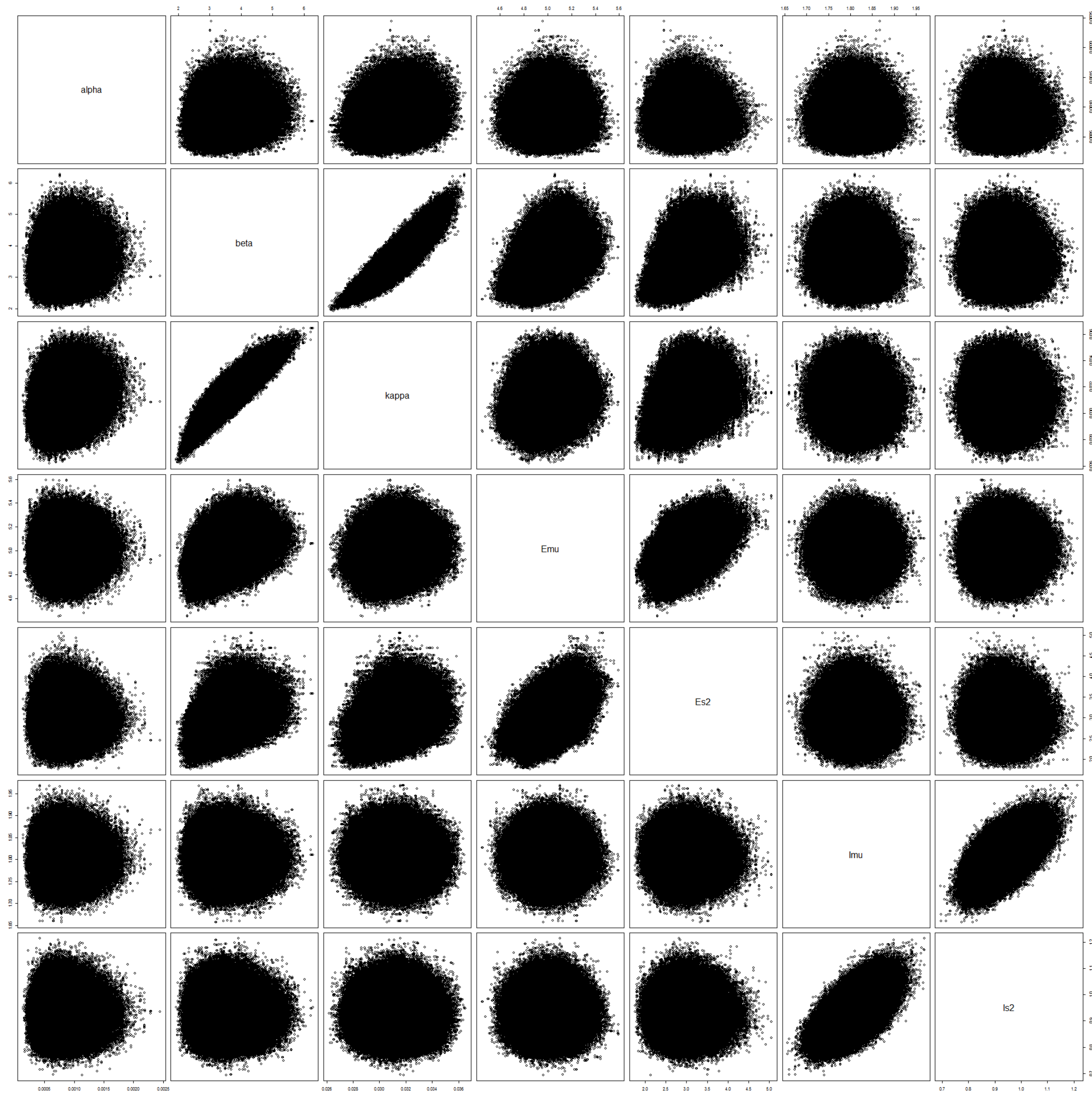

Figure 2.4.4: Pair plots for the RJMCMC samples of the posterior distributions of the model parameters.

Chapter 3

Model selection techniques - a Review of Existing Methods

The fitting of epidemic models has presented several interesting challenges in model comparison. A feature typical of epidemic data is a lack of replication of observations, that is, the epidemic process is only observed once. Since an epidemic consists of a series of infection events which are not independent nor identical, an epidemic cannot be taken as a realisation of multiple independent identically distributed events. As a result, the parameter posteriors are more likely to be non-normal (for example, [67]) leading to difficulties in choosing point estimators for measures of model fit such in the case of the DIC ([164], see Section 3.2 on page 57, and [69] for more discussion).

The partial nature of the observations in epidemic settings also leads to further challenges in determining model fit. Data augmentation (described in Subsection 2.3.3 on page 27) can be used to treat missing data as a nuisance parameter, and hence fit the model within the Bayesian framework using Data Augmented MCMC. However, as a result the parameter posteriors are sensitive to the choice of parameter priors. This creates challenges in using methods which use information from the parameter posteriors (for example, the DIC). Some measures of model fit may be sensitive to choice of parameter prior, even if the parameter posteriors are insensitive to the parameter priors.

In addition, epidemic models can have a high level of complexity, with different varieties of compartmental models that can be used, and different sources of population heterogeneity that can be modelled, for example species (for example, [36, 100]), households (for example, [104]), vector-based transmission and seasonality (for example, [95]), etc. mis-specification of any of these can be a source of model misfit. A measure of model fit must thus focus on sources of model inadequacy that are relevant to the purpose of the model; that is, one must "worry selectively about model inadequacies" [24].

There have been a variety of approaches at developing methods for assessing the fit of epidemic models to the data. These can be organised on a spectrum, consisting of purely Bayesian techniques at one end and the frequentist techniques at the other end. In the middle are techniques combining both Bayesian and frequentist techniques. In this chapter, we present an overview of the existing techniques in the literature to assess the fit of an epidemic model. First, we cover Bayesian model choice methods. Next an overview of posterior predictive checking methods is given. Finally, an overview of Deviance Information Criterion in its various forms is given.

3.1 Bayesian model choice methods

Within the context of two competing models M_1 and M_2 and observed data y, let $\pi(\theta_i|M_i)$ for $i \in \{1,2\}$be the parameter prior for θ_i, the parameters of model i. Let p_i be the prior probability of M_i, then the marginal likelihood of $y|M_i$, also known as the evidence:

$$\Pr(y|M_i) = \int \pi(y|\theta_i, M_i)\pi(\theta_i|M_i)d\theta_i$$

Then by Bayes theorem, for $i \in \{1,2\}$, we have:

$$\Pr(M_i|y) = \frac{\Pr(y|M_i)\Pr(M_i)}{\Pr(y|M_1)\Pr(M_1) + \Pr(y|M_2)\Pr(M_2)}$$

Hence

$$\begin{aligned}\frac{\Pr(M_1|y)}{\Pr(M_2|y)} &= \frac{\Pr(y|M_1)\Pr(M_1)}{\Pr(y|M_2)\Pr(M_2)} \\ &= \frac{\Pr(y|M_1)}{\Pr(y|M_2)} \cdot \frac{\Pr(M_1)}{\Pr(M_2)}\end{aligned} \tag{3.1.1}$$

The *Bayes factor* of M_1 vs M_2, denoted B_{12} is defined as:

$$B_{12} = \frac{\Pr(y|M_1)}{\Pr(y|M_2)}$$

which using Equation 2.3, can also be expressed:

$$\begin{aligned}B_{12} &= \frac{\frac{\Pr(M_1|y)}{\Pr(M_2|y)}}{\frac{\Pr(M_1)}{\Pr(M_2)}} \\ &= \frac{\text{Posterior odds of Model 1 vs. Model 2}}{\text{Prior odds of Model 1 vs. Model 2}}\end{aligned}$$

If the value of this ratio is above a certain level, [92], [99] then there is considerable support in favour of M_1. Note that in contrast with classical hypothesis testing, the two models are given equal status, as opposed to the frequentist framework where one assigns a "null" and "alternative" hypothesis, of which the null can never be accepted, but only "fail to be rejected".

The above representation of the Bayes factor as the ratio of the posterior odds to the prior odds shows that the prior odds of Model 1 vs. Model 2 affects the value of the Bayes factor, and therefore the decision on which hypothesis to accept. In the situation where there is a lot of prior information, this can be incorporated into the prior odds.

A related way of ranking models is through the use of posterior model probabilities. This was first put forward by Draper [47] who suggested an approach to model selection as follows: consider the model (S, θ) where θ are the model parameters and S is the structural assumptions of the model. The Bayesian approach is to calculate the posterior distribution of the model. However, calculation of this integral over the set of all possible models may be impossible since the space of all models is so large that it would not be possible to use a diffuse prior. Draper illustrates this in [47] by considering the following example:

Example 9. Suppose the observed data $Y = (Y_1, Y_2, \ldots, Y_n)$ is a sequence of binary numbers.

Each model for the data is a joint distribution for all possible observations:

$$p_{0000\ldots0}, p_{1000\ldots0}, p_{0100\ldots0}, p_{1100\ldots0}, \ldots, p_{1111\ldots1}$$

Thus, the set of all models is

$$\begin{aligned}\{(&p_{0000\ldots0}, p_{1000\ldots0}, p_{0100\ldots0}, p_{1100\ldots0}, \ldots, p_{1111\ldots1}) : \\ &p_{0000\ldots0}, p_{1000\ldots0}, p_{0100\ldots0}, p_{1100\ldots0}, \ldots, p_{1111\ldots1} \leq 1, \\ &p_{0000\ldots0} + p_{1000\ldots0} + p_{0100\ldots0} + p_{1100\ldots0} + \ldots + p_{1111\ldots1} = 1\}\end{aligned}$$

Different structural assumptions of the model, for example, independence, or identical distribution etc. would correspond to different subspaces of the model space. The dimension of the model space is $2^n - 1$. The dimension of the model space increases exponentially as the size of the data increases. For example, for a data size of $n = 10$, the model space would be of dimension 1023. Thus, it would be unreasonable to expect that each additional observation would add sufficient amounts of information about the relative plausibility of the various structural choices.

Thus, Draper suggests starting with a candidate model and performing what is called "model expansion", in which, starting from the candidate model the space of models is expanded to include other likely models.

This could be in some way related to the full posterior model distribution in that this distribution is the same distribution if zero prior density is assigned to certain sets of models.

Model expansion can either be continuous or discrete. Continuous model expansion involves embedding the model into a larger continuous class of models, the initial model being a special case of the larger class of models, for example, the Unif$(0,1)$ distribution as a special case of the Beta(α,β), distribution (since Unif$(0,1) \sim$ Beta$(1,1)$), or the Exponential as a special case of the Gamma distribution (since if $X \sim \text{Exp}(\lambda)$, then $X \sim \text{Gamma}(1,\lambda)$). Discrete model expansion involves the expansion of the model into a discrete class of models, where a finite or countably infinite number of models are compared. Suppose that we would like to compare several competing epidemic models $M_1, M_2, M_3, \ldots, M_k$, with parameter vectors $\theta_1, \theta_2, \theta_3 \ldots, \theta_k$ and prior distributions $\pi(\theta_1), \pi(\theta_2), \pi(\theta_3) \ldots, \pi(\theta_k)$. The posterior model probability can be obtained by using Bayes Theorem:

$$\Pr(M_j|y) \propto p_j \Pr(y|M_j) = p_j \int \pi_j(y|\theta_j, M_j)\pi(\theta_j|M_j)d\theta_j \tag{3.1.2}$$

where p_j is the prior probability of model $j \in \{1, 2, \ldots, k\}$. These posterior model probabilities can be used for model averaging.

There are several ways of calculating the Bayes factor, and the posterior model probabilities [99]. Bayes factors are usually difficult to calculate analytically, so tend to be calculated through the use of approximations or iterative methods such as [31] (a Gibbs sampler in which a model indicator variable is added, pseudo-priors are specified for the parameters not in each model, and MCMC is performed for the joint posterior of all models to be compared, their parameters and the model indicator) and RJMCMC [75] (similar to the previous algorithm, except rather than incorporating all the parameters of all the competing models and the model indicator into a large parameter vector, RJMCMC instead allows dimension switching moves between models). Such algorithms, which explore the model space, tend to be difficult to implement and tune for adequate mixing. The problems in using these algorithms within the context of epidemic model selection will be discussed in further detail later.

Readers should note the difference between Bayesian and classical approaches to model selection. In the classical Neyman-Pearson approach a null hypothesis model H_0 is compared against an alternative hypothesis model H_A. To assess model adequacy, a test statistic, a measure of the discrepancy between the data and the null hypothesis which also supports the alternative hypothesis is calculated. If the test statistic is within the critical region, which event should have a probability no greater than α, the specified type I error rate (recall a type I error is rejecting the null hypothesis even though the null hypothesis is true). If the test statistic is in this region, then the null hypothesis is rejected. If this is not the case, then the null hypothesis cannot be accepted, it can only be "failed to be rejected". From this, one can observe that the models are not given even status. In contrast, in the Bayes Factor/Posterior Model Probability approach, one specifies two (or more) candidate models. The amount of posterior support for the data being generated by each model is calculated. Hence, in summary, the classical method assesses model adequacy based on the probability of getting a test statistic more extreme than would be observed under the null hypothesis, whilst the Bayesian methods of model assessment selects models based on the posterior support that the data comes from that model. In certain circumstances this can lead to Bayesian tests and frequentist tests giving opposite results, as exemplified by the Jeffreys-Lindley paradox (see [153],[91],[112] for more information). A counter-intuitive property of Bayes factors is that Bayes factors can be over-conservative when used for model selection. That is, when using Bayes factors to select among is a set of models, these factors tend to favour the simpler models. To intuitively see this, consider the following example where two models, M_1 and M_2 are compared.

Example 10. Let model M_1 be parametrised by $\theta_1 = (\theta_{11}, \theta_{12}, \ldots, \theta_{1d_1})$, a vector of d_1 non-negative components. Let model M_2 be parametrised by $\theta_2 = (\theta_{21}, \theta_{22}, \ldots, \theta_{2d_2})$, a vector of d_2 non-negative components.

Suppose $d_2 > d_1$.

Let the prior distribution on the parameters θ_i of M_i be such that each component is i.i.d Unif$(0, A)$, that is,

$$\pi(\theta_i) = \begin{cases} \left(\frac{1}{A}\right)^{d_i} & \text{if all of } \theta_{i1}, \theta_{i2}, \ldots, \theta_{id_1} \in (0, A) \\ 0 & \text{otherwise} \end{cases}$$

Then:

$$\begin{aligned}\Pr(y|M_i) &= \int_{\Theta_i} \pi(y|\theta_i, M_i)\pi(\theta_i|M_i)d\theta_i \\ &= \left(\frac{1}{A}\right)^{d_i} \int_0^A \int_0^A \cdots \int_0^A \pi(y|\theta_i, M_i)d\theta_{i1}d\theta_{i2}\dots d\theta_{id_i}\end{aligned}$$

The evidence for M_1 tends to be larger than M_2 since $d_2 > d_1$, if the integral is similar in value for both models. To demonstrate this:

Consider the case where for some $K, a > 0$, $\pi(y|\theta_i, M_i) \leq Ke^{-a\sum\theta_{ij}}$ for any fixed y and $i = 1, 2$.

$$\begin{aligned}\Pr(y|M_i) &= \left(\frac{1}{A}\right)^{d_i} \int_0^A \int_0^A \cdots \int_0^A \pi(y|\theta_i, M_i)d\theta_{i1}d\theta_{i2}\dots d\theta_{id_i} \\ &\leq \left(\frac{1}{A}\right)^{d_i} \int_0^A \int_0^A \cdots \int_0^A Ke^{-a\sum\theta_{ij}}d\theta_{i1}d\theta_{i2}\dots d\theta_{id_i} \\ &= \left(\frac{1}{A}\right)^{d_i} K\left(1 - \frac{e^{-aA}}{a}\right)^{d_i} \\ &\leq \left(\frac{1}{A}\right)^{d_i} K\end{aligned}$$

Hence, as $A \longrightarrow \infty$, $\Pr(y|M_i)$ will tend to a fixed constant, which will tend to be larger for M_1 as $d_2 > d_1$.

A simple demonstration of this can be seen in the following example:

Example 11. We extend the example from [153], to consider the case where there is a sample of n observations from a normal distribution of known variance, $\forall i \in \{1, \dots, n\} : x_i \sim N(\theta, \sigma^2)$ (instead of a single observation). Set the prior to be $\theta \sim N(\theta_p, \sigma_p^2)$, and suppose that the hypotheses to be tested are:

$$\begin{aligned}H_1 &: \theta = \theta_0 \\ H_2 &: \theta \neq \theta_0\end{aligned}$$

H_1 is a model with no free parameters (dimension 0) whilst H_2 is a model with

1 parameter. Then we have:

$$\begin{aligned} B_{12} =& \frac{\phi\left(\frac{\bar{x}-\mu_0}{\frac{\sigma^2}{n}}\right)}{\int \phi\left(\frac{\bar{x}-\mu}{\frac{\sigma^2}{n}}\right)\phi\left(\frac{\mu-\mu_p}{\sigma_p^2}\right)d\mu} \\ =& \left(1+\frac{n\sigma_p^2}{\sigma^2}\right)^{\frac{1}{2}} \frac{\exp\left\{\frac{1}{2}\left(\sigma_p^2+\frac{\sigma^2}{n}\right)^{-1}(\bar{x}-\mu_p)^2\right\}}{\exp\left\{\frac{1}{2}\frac{n}{\sigma^2}(\bar{x}-\mu_0)^2\right\}} \end{aligned}$$

Then as $\sigma_p^2 \longrightarrow \infty$, $B_{12} \longrightarrow \infty$, regardless of the observed data. Note that this arises primarily from the fact the H_2 is a more complex model (it has one more parameter) and thus there is one more parameter to integrate over.

In fact the Bayes factor is bounded from below:

$$\begin{aligned} B_{12} = \left(1+\frac{n\sigma_p^2}{\sigma^2}\right)^{\frac{1}{2}} \frac{\exp\left\{\frac{1}{2}\left(\sigma_p^2+\frac{\sigma^2}{n}\right)^{-1}(\bar{x}-\mu_p)^2\right\}}{\exp\left\{\frac{1}{2}\frac{n}{\sigma^2}(\bar{x}-\mu_0)^2\right\}} \geq& \frac{\exp\left\{\frac{1}{2}\left(\sigma_p^2+\frac{\sigma^2}{n}\right)^{-1}(\bar{x}-\mu_p)^2\right\}}{\exp\left\{\frac{1}{2}\frac{n}{\sigma^2}(\bar{x}-\mu_0)^2\right\}} \\ \geq& \frac{1}{\exp\left\{\frac{1}{2}\frac{n}{\sigma^2}(\bar{x}-\mu_0)^2\right\}} \end{aligned}$$

This disadvantage of the Bayes factor makes it unsuitable for use unless there is strong prior information available.

If there is prior information available, problems can naturally arise when this prior information is inappropriate or misleading. This is because the posterior model probabilities and Bayes factors are more sensitive to the priors on the parameters than the parameter posterior distribution is sensitive to the priors on the parameters . Reference [189] gives an example which shows this prior sensitivity intuitively. Consider a model with a single parameter θ_j, where all of the likelihood is negligible outside the interval $[0, 1]$. If the uniform prior $\text{Unif}(-100, 100)$ is used, suppose the marginal likelihood $\Pr(y|M_j)$ is b. That is:

$$\Pr(y|M_j) = \int \pi(y|\theta_j, M_j)\pi(\theta_j|M_j)d\theta_j = b$$

If the prior is changed to $\text{Unif}(-1000, 1000)$, the parameter posteriors will not change significantly, but the marginal likelihood $\Pr(y|M_j)$ will essentially decrease by a factor of 10 to $\frac{b}{10}$. Essentially, the act of integrating out the parameters makes the Bayes factor and the posterior model probabilities more sensitive to choice of parameter prior distribution than the parameter posterior distributions is sensitive to the choice of parameter prior distribution.

As mentioned earlier, the posterior model probabilities and Bayes factors have been frequently calculated using Reversible Jump MCMC (RJMCMC), which can be used to produce a Markov chain which has a limiting distribution which is that of Equation 3.1.2. Suppose we have a set of models $M_1, M_2, M_3, \ldots$which have parameter vectors , which may be of different dimensions (for $k \in \mathbb{N} : \boldsymbol{\theta}_k$ is the parameter vector for M_k). A model can also represent a candidate model that we are selecting from, for example a spatial kernel. Instead of the usual state vector, we take $(k, \boldsymbol{\theta}_k)$, where $\boldsymbol{\theta}_k \in \mathbb{R}^{n_k}$ as the current chain state. The algorithm is as follows:

1. Propose the move to model $k^{'}$ from k, from model proposal distribution $j(k'|k)$. The dimension of the parameter spaces of the two models may be different.

2. Let $\boldsymbol{u}$ and $\boldsymbol{u}'$ be vectors of length r and r' such that $n_k + r = n_k^{'} + r^{'}$. Sample $\boldsymbol{u}$ from a proposal distribution with joint density $g(\boldsymbol{u})$ (the probability density of the reverse move is $g^{'}(\boldsymbol{u}^{'})$).

3. Generate a proposed $\theta_{k'}^{'}$ from the diffeomorphism $(\boldsymbol{\theta}_{k'}^{'}, \boldsymbol{u}') = h(\boldsymbol{\theta}_k, \boldsymbol{u})$.

4. Accept the move $(\boldsymbol{\theta}_{k'}^{'}, \boldsymbol{u}')$ with probability

$$\alpha\left((\boldsymbol{\theta}_{k'}^{'}, \boldsymbol{u}')|(\boldsymbol{\theta}_k, \boldsymbol{u})\right) = \min\left\{1, \frac{\pi(k^{'}, \boldsymbol{\theta}_{k'}^{'}|\boldsymbol{x})}{\pi(k, \boldsymbol{\theta}_k|\boldsymbol{x})} \frac{j(k|k^{'})}{j(k'|k)} \frac{g^{'}(\boldsymbol{u}^{'})}{g(\boldsymbol{u})} \left|\frac{\partial(\boldsymbol{\theta}_{k'}^{'}, \boldsymbol{u}')}{\partial(\boldsymbol{\theta}_k, \boldsymbol{u})}\right|\right\}.$$

The posterior model probability is estimated as the proportion of iterations spent by the Markov chain in the parameter space of each model. Note that the data augmented MCMC algorithm (in Section 2.3.3) is a special case of the RJMCMC algorithm, where the Jacobian determinants are simple to compute.

The unobserved data that is often present in epidemic data makes it difficult to apply the Bayes factor or posterior model probability approaches to the comparison of spatial kernels as it is difficult to develop computer algorithms that will allow mixing of the RJMCMC – to transition between models would involve transition between spatial kernels. Since the unobserved infection times are dependent on the spatial kernel the chain would have a low acceptance rate for proposals to move between models with different spatial kernel. This is because RJMCMC requires the specification of a mapping to move between the different parameter spaces of each model. Given that the infection times (the augmented data) between each model are not equivalent to each other, it is non-trivial to specify a mapping h to move between the states in two epidemic models which produces a satisfactory acceptance probability.

Furthermore, there is the added difficulty of there being cryptic infections. The complexity of using RJMCMC on the epidemic models is compounded by the fact that RJMCMC is used to fit each model of an epidemic. In this situation RJMCMC is used to move between different dimensional parameter spaces within the same model, which correspond to different numbers of cryptic infections. Hence, each model can be defined for the purposes of RJMCMC as a set of models, each representing an epidemic with different numbers of cryptic infections, so the full model space is actually a product space of the candidate models with all the models representing different levels of cryptic infections. The structure of the compartments in each of the candidate models to be compared may be quite different. For example, consider an SEIR model versus a SIR model, where the unobserved state in the former would be E and I in the latter. The unobserved data in each model is different, making it difficult to specify a mapping h to allow the RJMCMC algorithm to move between model spaces. Hence, the problem of determining an algorithm which would mix well for this general problem is not a trivial one.

Nevertheless, there have been several attempts to use Bayesian model choice for epidemic model selection. An example of such a paper is the paper by [131], which examines the 1861 Hagelloch measles epidemic. This data-set has a particularly large amount of information about the population, comprising each individual's name, age, sex, time of appearance of first symptoms, and the date the rash first appeared as well as other information. The model itself represents the following process for each individual i: the appearance of symptoms (at time S_i), appearance of rash (at time Q_i), time of infection I_i, and time of removal R_i. The time of first appearance of symptoms is typically the time of appearance of Koplik spots (white spots on the inside of the cheek). The time of the appearance of the rash refers to the time of appearance of the red skin rash. The force of infection is modelled as being related to a household effect β_H, classroom effects β_C^1, β_C^2 (L_i denotes the classroom of individual i), and a distance effect β_G (distance between individuals i and j is $\rho(i,j)$).

$$\alpha_{ij} = \beta_H \mathbf{1}_{\{\rho(i,j)\}} + \beta_C^1 \mathbf{1}_{\{L_i = L_j = 1\}} + \beta_C^2 \mathbf{1}_{\{L_i = L_j = 2\}} + \beta_G \exp\left(-\theta\rho(i,j)\right)$$

First, the authors compared each of the models that exclude one household or classroom effect and found that the rank of the models was relatively robust to the specification of the spatial kernel for both fixed and imputed infection times; but no Bayesian comparison was made of the spatial kernels. Rather, they noted that the posterior model probabilities gave similar model rankings with other choices of spatial kernel chosen, and thus that the model rankings were robust to choice of spatial kernel.

Note that in all models that were compared, the nature of the missing data, that is, the infection and the removal times, were the same. Therefore, RJMCMC only requires to formulate dimension changing moves related to model parameters but not the augmented data, simplifying the algorithm and its implementation. Runs with simulated data-sets found that whilst, in general, the correct model was selected, model ranking was affected by prior specification, with more informative priors yielding more posterior support for the full model in some simulated data-sets. On epidemics generated with models with no household effects, there was difficulty in identifying the correct model, as the household effect can be compensated for by the kernel parameters and the spatial coefficient. This identifiability issue makes prior specification have influence over model ranking in this situation, making prior specification an important issue.

Another paper utilising Bayesian model choice is [104]. This paper uses an epidemic model which assumes the population contains a number of known individuals and is partitioned into several known households. The observed data consists of the final numbers of individuals in each household that have ever become infectious during the epidemic. All individuals are assumed to be susceptible at the start of the epidemic. More formally, if there are n_{ij} households containing j individuals that are initially susceptible of whom i become infected,

$$D = \left\{ n_{ij}, j = 1, 2, \ldots, i = 0, \ldots, j \right\}$$

The model that is used in the paper assumes that a given infective makes contact with the global population at times given by a Poisson process of homogeneous rate λ_G and the contact is randomly selected from the population. There is also a local infection rate λ_L, where within the household contacts are made at times given by a homogeneous Poisson process with rate $n\lambda_L$ (where n is individual's household size), and the individual that makes contact is selected randomly from the individuals and household. An assumption of mutual independence is made regarding all the Poisson processes. The epidemic ends when the whole population is infected. The infectious period is assumed to be constant, and the final outcome distribution can be shown to be invariant to latent period, as long as the latent period is almost surely finite. The paper tests between three competing models:

1. M_1 with parameters λ_L and λ_G
2. M_2 where $\lambda_L = \lambda_G$
3. M_3 where $\lambda_L = 0$

The authors use exponential priors for the model parameters. In this paper, the augmented data consists of the infectious contacts, both local and global, and the recipients of these contacts. Thus, the RJMCMC algorithm does not have to make moves with respect to the augmented data, only with respect to the parameters $\theta = (\lambda_L, \lambda_G)$. This eliminates some the potential problems associated with using RJMCMC in epidemic models of this kind.

Regarding prior sensitivity, the authors present a lemma which shows that if there is at least one partially infected household the Bayes factor will favour M_2 and M_3 which are the simpler models as the priors become increasingly vague. Quoting from the paper [104]:

Lemma 12. *Case 1, partially infected households: If there exists at least one household with at least one, but not all, members infected, then* $\lim_{\mu \to 0} B_{12}(\mu) = \lim_{\mu \to 0} B_{13}(\mu) = 0$.

Case 2, fully infected households in a partially infected population: Suppose that not all households are infected, but that in every infected household, all members of that household are infected. If all infected households are of size one, then $\lim_{\mu \to 0} B_{12}(\mu) = \lim_{\mu \to 0} B_{13}(\mu) = 1$. *Conversely, if there exists at least one infected household of size two or more,* $\lim_{\mu \to 0} B_{12}(\mu) > 1$ *and* $\lim_{\mu \to 0} B_{13}(\mu) > 1$.

Case 3, fully infected population: If every individual in the population is infected, then $\lim_{\mu \to 0} B_{12}(\mu) = \lim_{\mu \to 0} B_{13}(\mu) = 1$.

Consider a population in which all households become infected apart from one household which is only partially infected with a single individual becoming infected by the end of the epidemic. In this case, this will fulfil Case 1 of the lemma, which would favour M_2 and M_3 very heavily, as the priors become increasingly vague. If this individual does not get infected, we would have Case 2 where there will be a preference for M_1, as the priors become increasingly vague. If this whole household is infected, then we would have equal weight on M_1 versus M_3, as the priors become increasingly vague. There would also be equal weight on M_1 versus M_2, as the priors become increasingly vague. Hence, as the priors become more uninformative, which model is favoured can depend on the outcome of a single household. This shows the importance of prior specification in Bayesian model selection.

3.2 Deviance Information Criterion

Another measure used for model comparison is the deviance information criterion, also known as the DIC. This measure was first proposed by [164], basing the choice of measure through approximate decision theoretical reasons, using a logarithmic utility function. The DIC for observed data, also known as the observed DIC, denoted DIC_1, can be expressed:

$$DIC_1 = -4E_\theta \left\{ \log \pi(y|\theta)|y \right\} + 2 \log \pi(y|\tilde{\theta})$$

where $\tilde{\theta}$ is an estimate of θ.

DIC_1 can also be expressed as:

$$DIC_1 = -2E_\theta \left\{ \log \pi(y|\theta)|y \right\} + P_D$$

where $P_D = -2E_\theta \left\{ \log \pi(y|\theta)|y \right\} + 2 \log \pi(y|\tilde{\theta})$. The first of the terms of the DIC was proposed in the original paper [164] as a measure of model goodness of fit, and the second term as a measure of model complexity. The original paper [164] proposed this measure for model selection with a heuristic justification for the measure.

To compare models using the DIC each candidate model is fitted to the data and the DIC is calculated conditioning on the model that has been fitted. The DICs are then compared and the models are ranked by DIC with the model with the lowest DIC being ranked the highest or most adequate model. This method of model selection has the benefit that it is easily integrated into MCMC, and is implemented in the software package WinBUGS. However, there are some disadvantages with the method: note that P_D is not invariant to the estimate used for $\tilde{\theta}$. Some may argue that this approach is fundamentally not a Bayesian approach (see [69] and discussions on the papers [164, 165], for example, the discussion by Dawid on the former). The DIC for each model is calculated conditioning on that model so there is a lack of a “standard perspective” from which to compare models and interpret jointly the DICs for two mutually exclusive models.

As stated earlier, the justification for the DIC given in the original paper [164] is a heuristic one. The key idea behind the DIC is the idea of model parsimony: finding the optimal trade-off between model fit and model complexity, model complexity being represented by effective number of parameters P_D. The authors of the original paper [164] using idea called "focus", where the parameters of interest in a hierarchical model must be identified before determining the form of the DIC. Suppose a hierarchical model is to be assessed in which the parameters of the model are θ_1 which depend on a further set of parameters θ_2. Depending on how the model is to be used, the focus can either be θ_1 or θ_2 and thus $\theta = \theta_1$ or $\theta = \theta_2$ or $\theta = (\theta_1, \theta_2)$ depending on what the parameters of interest are. Whilst the idea of “focus” in model selection is intuitive for simple models, in more complex models, such as hierarchical models, mixture models, and models with missing data there are many different approaches regarding the treatment of the missing data, the parameters, and estimate $\tilde{\theta}$ used for the parameters leading to different forms of the DIC. This was investigated in [34] where the authors of this paper investigated various forms of the DIC for mixture models and missing data models. They proposed eight different forms of DIC, depending on how the missing data and the parameters is treated as well as the form of the parameter estimate. These include different forms for DIC when the likelihood can be derived analytically for observed data, and where it is not possible.

Each of the proposed forms of the DIC give different model rankings. In fact, there are infinitely many possibilities for the forms of DIC, depending on how “missing” data are specified, giving infinitely many different model rankings. As a result, the selection of DIC to use in model assessment needs to be carefully considered. This is especially true in epidemic modelling, which often requires the use of one of the missing data DICs, in which there are many different ways that the focus and estimator can be specified, with no clear way to select which is the most appropriate form of DIC.

In the comments to the paper by Celeux [126], it is also noted that the augmented data z may be of high dimension, and that whilst the observed data may contain enough information to estimate $\hat{\theta}$, there might not be adequate information to produce a valid estimate of the augmented data z.

Reference [43] compares several versions of the DIC (DIC_1 with the posterior mean, median and maximum as $\tilde{\theta}$, DIC_3, and a form of the DIC from [59]) for spatial temporal models in discrete time with no latent period, but with latent susceptible classes. Using simulated data they find DIC_3 to be more robust to lack of information on the susceptible latent classes.

In the context of spatial temporal epidemic modelling, as shown in the earlier sections the likelihood of the observed data $\pi(y|\theta)$ cannot usually be found analytically. Thus, the forms of the DIC that have been applied in epidemic modelling are usually the missing data variants. Several examples of DICs that have been used to compare epidemic models are:

$$DIC_4 = -4E_{\theta,x}\left\{\log \pi(y,x|\theta)|y\right\} + 2E_x\left\{\log \pi(y,x|E_\theta(\theta|y,x)|y\right\}$$

$$DIC_6 = -4E_{\theta,x}\left\{\log \pi(y,x|\theta)|y\right\} + 2E_x\left\{\log \pi(y,x|\hat{\theta}(y))|y\right\}$$

where $\hat{\theta}(y)$ is an estimate of θ from the observed data posterior distribution $\pi(\theta|y)$, and

$$DIC_8 = -4E_{\theta,x}\left\{\log \pi(y,|x,\theta)|y\right\} + 2E_x\left\{\log \pi(y,|x,\hat{\theta}(y,x)|y\right\}$$

where $\hat{\theta}(y,x)$ is an estimate of θ from the observed data posterior distribution $\pi(\theta|y,x)$.

In [104] several epidemic models were compared using the DIC_4. For some of the models and datasets compared, the DIC was found to be less prior sensitive than the Bayes factor in ranking models. In [111] DIC_4 and DIC_8 was used to rank spatio-temporal SEIR models with different spatial kernels. This was done with both real world and simulated data. The model rankings produced were different depending on the form of DIC used.

Thus, since different forms of the DIC yield different model rankings it is important to be as clear as possible about what decisions are being made when choosing a form of the DIC. The focus of the DIC is particularly important. In the data augmentation methods which are often used for epidemic models, the choice of latent processes embedded in the marginal model makes no difference to the posterior distributions of the parameters of interest. However, with the DIC, the choice of latent process and whether it is treated as missing data or as a parameter of interest affects the model rankings. Several of the missing data DICs from [34] can be expressed as the posterior mean of some measure calculated by a notional observer of the complete data. In the case of DIC_4 we can express this as:

$$\begin{aligned} DIC_4 &= E_x[-4E_\theta\{\log \pi(y,x|\theta)|y,x\} + 2\log \pi(y,x|E_\theta(\theta|y,x)|y] \\ &= E_x[DIC_1(x,y)] \end{aligned}$$

that is, the expectation over x (the full data) of $DIC_1(x,y)$, the DIC_1 of the observed and unobserved data. This can be interpreted as a Bayesian observer's posterior mean (given the observed data y) of DIC_1 computed over the full data x, and is a natural extension of DIC_1 to missing data. DIC_6 can be expressed as

$$DIC_6 = E_x[-4E_\theta\{\log \pi(y,x|\theta)|y,x\} + 2\log \pi(y,x|\hat{\theta}(y))|y]$$

where the estimate $\hat{\theta}(y)$ is only based on the observed data and not the full data x. In this DIC, the latent observer, despite having access to the full data, bases his/her estimate on the observed data y only. This seems less rational than in DIC_4. DIC_8 can be written as:

$$DIC_8 = E_x[-4E_\theta\{\log \pi(y|x,\theta)|y,x\} + 2\log \pi(y|x,\hat{\theta}(y,x))|y]$$

where the estimate $\hat{\theta}(y,x)$ is based on the observed data and the full data x, but only the partial likelihood $\pi(y|x,\theta)$ of the observed data is used. Hence, the focus of this DIC is on the model for y conditional on x. In some cases, DIC_8 may not have an appropriate focus, for example the case where the data x is observed through an observation process, yielding the observed data y, such that the model can be expressed:

$$\pi(x,y|\theta) = \pi(x|\theta_1)\cdot\pi(y|x,\theta_2)$$

Where θ_1 is the parameter vector for the dynamical model of x, and θ_2 is the parameter vector for the observation process, which gives the observed data y from the full data x. In this case, since the aspect of the model that is of interest is the dynamical model for x, it is clear that the focus of DIC_8 is inappropriate, as its focus is on the observation model for y given x, $\pi(y|x,\theta_2)$. In [111], as well as using posterior predictive p-values, $DIC_8$was used to assess the adequacy of spatial kernels in models fitted to Giant Hogweed data. The models assessed were SI models, which were fitted to snapshot data. This is an example of such a model described earlier, with the SI model being the dynamical model for the full data x, where θ_1 is the parameter vector of the parameters for the SI model, and observation model being parametrised by θ_2, the probability at each snapshot of reporting a site as colonised given that it has been colonised. Because of this, DIC_8 may not be the most appropriate choice of model comparison measure.

In summary, there are several properties of the DIC which make it attractive for model comparison. The calculation of the DIC fits well into the data augmented RJMCMC methods used to fit epidemic models, hence the calculation of the DIC is far more straightforward than with Bayes factors. This is primarily because only one model needs to be fitted at one time, so complicated algorithms do not need to be used to jump between competing epidemic models – as mentioned earlier, when comparing spatial kernels, a model in the sense of epidemic modelling is actually a class of models in terms of RJMCMC and is not trivial to find an algorithm to move between models. The motivation for the DIC is straightforward for simple cases: the idea of parsimony in combination with the ability to explain the data, is an intuitively natural way to rank models.

However, there are several disadvantages to the DIC. The DIC for each model is calculated conditioning on that model, so there is no unified single Bayesian observer from whose perspective the models are compared. This makes it difficult to interpret the model rankings given by the DIC. In addition, with missing data, the idea of "focus", which is straightforward with simple models, leads to many different forms of DIC. In addition, since the posteriors of epidemic models are often non-normal, it is not clear what should be used as an estimator for the parameters. Different forms of the DIC lead to different rankings and it is not clear which choice of DIC is optimal in any given situation. In addition to the points above, [165] summarised the many criticisms of the DIC which occur even when there is no unobserved data and the models are relatively simple: the DIC is not invariant to re-parameterisation, that the underlying philosophy of the DIC does not include the belief of a "true model" (making the results of such comparisons difficult to interpret). The DIC was created to assess whether a model can produce replicate data consistent with the observed data, yet uses a point estimate instead of the full posterior predictive distribution. The justification behind the DIC was heuristic, and there is no rigorous theoretical justification behind the DIC.

3.3 Posterior predictive checking

Another approach to assessing model fit is to check the discrepancy between the predictions made by the model and the observed data [78]. If a model is used to create predictions, and if decisions are made based upon these predictions, it is important that these predictions are realistic. A model may be "incorrect" as long as it does not affect the predictions in any substantial way as far as decision-making is concerned. In frequentist statistics, this can be done through the use of a statistical test, which produces a p-value. In Bayesian statistics this is often done by checking observed quantities against a reference distribution.

One method developed for checking observed data against a reference distribution was proposed by [25], who proposed that the prior predictive distribution, which does not require any data to be observed, to be used as a reference distribution:

$$\pi(y) = \int_{\Theta} \pi(y|\theta)\pi(\theta)\, d\theta$$

In his paper, Box stated "I believe that it is impossible logically to distinguish between the model assumptions and the prior distribution of the parameters. The model is the prior in the wide sense that it is a probability statement of all assumptions currently to be tentatively entertained a priori. On this view, traditional sampling theory was of course not free from assumptions of prior knowledge.". By comparing the observed data using a "checking function", the amount of discrepancy from the model that has been assumed could be quantified, and if it were over a certain level, the model would be deemed inadequate. A natural choice of checking function is an analogue of the classical p-value:

$$p = P(T(y^{rep}) \geq T(y)) = \int P\left(T(y^{rep}) > T(y)|\theta\right) \pi(\theta) d\theta$$

where $y_{rep}$is a replication of the data (data from a replicate experiment), and T is a function known as a test statistic.

This is known as a *prior predictive* p-value. There are obvious downsides to this approach, as for example, uninformative improper prior distributions cannot be used. The results of a prior predictive check are heavily dependent on the prior distribution that is chosen; the model $\pi(y|\theta)$ may be rejected because of poor choices of prior.

Another method of checking model fit is the use of posterior predictive checking, which uses the *posterior predictive distribution* $\pi(y^{rep}|y)$ [78, 157, 158]:

$$\begin{aligned}\pi(y^{rep}|y) &= \int_{\Theta} \pi(y^{rep}|\theta, y)\pi(\theta|y)\, d\theta \\ &= \int_{\Theta} \pi(y^{rep}|\theta)\pi(\theta|y)\, d\theta\end{aligned}$$

where y^{rep} represents the data from a replicate experiment generated under the model. It is possible to use vague or improper priors for this method, as long as the posterior distribution is proper. The central idea behind this method of checking model fit is that the observed data y should not appear extreme when benchmarked against the posterior predictive distribution $\pi(y^{rep}|y)$.

The simplest type of model checking which uses the posterior predictive distribution is a graphical check [58], which involves computing some summary statistics from replicate data from the posterior predictive distribution and comparing that with the observed data graphically.

As with prior predictive checking, the statistical testing approach can be taken as well. Observed data can be compared to data obtained from the posterior predictive distribution via a checking function. A natural choice of checking function would be [125, 60]:

$$\begin{aligned} p(y) &= P(T(y^{rep}) > T(y)|y) \\ &= \int P\left(T(y^{rep}) > T(y)|\theta\right) \pi(\theta|y)d\theta \\ &= \int p(T, y, \theta)\pi(\theta|y)d\theta \end{aligned}$$

where $p(T, y, \theta) = P\left(T(y^{rep}) > T(y)|\theta\right)$. This checking function that is known as the posterior predictive p-value, which is the posterior mean of the probability of obtaining a test statistic greater than that with data generated under the current model. This can be interpreted as the posterior expected value of $p(T, y, \theta)$, the p-value of a classical test, that is, the posterior probability of obtaining a test statistic more extreme than would be observed under the null hypothesis, where the null hypothesis model is the current model. However, unlike the frequentist p-value, the prior distribution of the posterior predictive p-value is less stochastically variable than Unif$(0, 1)$ [125]. This can also be seen using the law of total variance.

$$\begin{tikzcd} y \arrow[r] & \pi(\theta|y) \arrow[r] & \pi(p(T,y,\theta)|y) \end{tikzcd}$$

Figure 3.3.1: Diagram of interpretation of the posterior predictive p-value

Since:

$$\text{Var}(p) = E_y(\text{Var}(p|y)) + \text{Var}_y(E(p|y))$$

then,

$$\begin{aligned}\text{Var}_y(E(p|y)) &= \text{Var}(p) - \underbrace{E_y(\text{Var}(p|y))}_{\geq 0} \\ &\leq \text{Var}(p)\end{aligned}$$

Because of this, it is often better to look at the distribution of $p(T, y, \theta)$, which can be interpreted as the posterior predictive distribution of the classical p-value. This can be interpreted to obtaining a posterior distribution for the replicate data given the observed data, and handing this over to an independent frequentist observer who tests the fit of the model using a traditional frequentist test (see fig. 3.3.1).

There are several advantages to using posterior predictive p-values to test model fit.

1. First of all, note that no specific alternative hypothesis is needed to be specified. Unlike Bayes factors and the DIC, which compare between models, PPP-value can be used to assess model adequacy, in addition to the uses in model comparison.

2. Vague or improper priors may be used as long as the posterior distribution is proper. Bayes factors, as mentioned previously in section 3.1 on page 49, can favour one model over another when vague priors are used, and cannot be used when improper priors are used.

3. The posterior predictive p-value is sensitive to the choice of prior distribution

only to the extent that the posterior distribution is sensitive to the prior. This is contrast to the pure Bayesian approach in which the posterior model probabilities are more sensitive to the parameter prior distributions than the parameter posterior distributions (see section 3.1).

4. The calculation of posterior predictive p-values can be easily integrated within MCMC. This is because the data are used to obtain both the value of the test statistic and the posterior predictive distribution which is used to calculate the tail probability.

5. Unlike the DIC, the posterior predictive p-value does not need selection of a point-estimator for θ.

Hence, the specification of test statistic is of particular importance. In theory, any test statistic can be used. But in practice, if the test statistic is not chosen with care, the test will be unable to detect discrepancy with the null hypothesis. For example, [45] considers a sample from a normal distribution with unknown mean. An inappropriate choice of test statistic is the sample mean: in fitting the model to the data, the location parameter posterior will have a large amount of mass centred around the observed sample mean, and the posterior predictive distribution will therefore generate replicate data with a sample mean close to the observed sample mean. Hence, this test statistic is highly unlikely to detect any discrepancy between the predicted data and the observed data, and is an inappropriate choice of test statistic. More formally (writing out the verbal description in [45] in mathematical terms):

Example 13. Suppose that $Y = (Y_1, Y_2, \dots, Y_n)$ is an i.i.d. random sample. Suppose the model that is fitted to this data is Y_i i.i.d. with $Y_i \sim N(\theta, \sigma^2)$, θ unknown, σ^2 known. Suppose that a non-informative prior is used for θ, $\theta \sim N(\mu_0, \sigma_0^2)$ where μ_0 and σ_0^2 are specified.

The posterior distribution obtained is

$$\theta|y \sim N(\mu_1, \sigma_1^2)$$
$$\text{where } \sigma_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1}$$
$$\mu_1 = \sigma_1^2\left(\frac{\mu_0}{\sigma_0^2} + \frac{n}{\sigma^2}\bar{y}\right)$$

Consider the case when $T(y^{rep}) = \bar{y}_{rep}$. The posterior predictive distribution for this discrepancy statistic is:

$$\bar{Y}_{rep}|\theta \sim N(\theta, \frac{\sigma^2}{n})$$
$$\Rightarrow \bar{Y}_{rep}|y \sim N(\mu_1, \sigma_1^2 + \frac{\sigma^2}{n})$$

and thus the posterior predictive p-value is:

$$Pr(\bar{Y}_{rep} \geq \bar{y}|y) = 1 - \Phi\left(\frac{\bar{y} - \mu_1}{\sigma_1^2 + \frac{\sigma^2}{n}}\right)$$

Consider the case where there is little prior information,

As $\sigma_0^2 \to \infty$,

$$\sigma_1^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}\right)^{-1} \to \frac{\sigma^2}{n}$$
$$\mu_1 = \sigma_1^2\left(\frac{\mu_0}{\sigma_0^2} + \frac{n}{\sigma^2}\bar{y}\right) \to \bar{y}$$

Thus,

$$Pr(\bar{Y}_{rep} \geq \bar{y}|y) = 1 - \Phi\left(\frac{\bar{y} - \mu_1}{\sigma_1^2 + \frac{\sigma^2}{n}}\right) \to 1 - \Phi(0) = 0.5$$

This can be interpreted as the following: as the amount of prior information decreases, the discrepancy statistic can never find discrepancy between the data and the model. This is unsatisfactory.

However, one can choose instead a test statistic which quantifies discrepancy in regards to the variance, skewness or kurtosis of the observed data versus the replicated data. The reason why using the sample mean in this example cannot detect model mis-specification is that the data needs to be used twice, once to obtain the posterior distribution and again to obtain the tail probability. This leads to a *reinforcement* phenomenon, where the test tends to favour H_0 if the discrepancy measure is not chosen well.

As well as selecting a more appropriate choice of test statistic, there has been research on transforming the data (stripping the data of information) to lessen the double use of the data. In [14] the authors propose a conditional posterior predictive p-value as an alternative to the posterior predictive p-value. The conditional posterior predictive value is calculated in the following way:

1. Select a function U such that $U(X)$ and $T(X)$ share as little information as possible. Let the transformed observed data be denoted $u = U(y)$.

2. Calculate $t = T(x)$

3. Evaluate $\pi(\theta|u) \propto \pi(u|\theta)\pi(\theta)$ to obtain $\pi(\theta|u)$

4. The conditional posterior predictive p-value is:
$$p_{cpred}(y) = \int P\left(T^{rep} > t|\theta\right) \pi(\theta|u) d\theta$$

It appears straightforward to extend conditional posterior predictive p-values to models where data augmentation is used and also to include the use of discrepancy variables, by integrating over x and θ given y. However, it is difficult to choose a suitable U, and the posterior distribution $\pi(\theta|u)$ may only be obtainable through MCMC. In addition, it may be only possible to obtain $P\left(T^{rep} > t|\theta\right)$ through Monte-Carlo methods. Nested Monte-Carlo (Monte-Carlo within Monte-Carlo) greatly increases the computational burden and makes this approach of model assessment difficult to implement in practice.

Within the field of epidemic modelling, there have been several examples of posterior predictive testing using posterior predictive p-values for epidemiological data. In several papers, disease progress curves have been used for $T(y)$, for example to test the adequacy of spatio-temporal models for the Huanglongbing (HLB) virus in citrus plants in [145]. Such disease progress curves involve the predicted number of infectives at the times where such observations exist. If the number of infectives is not observed, for example in removal only data, the predicted number of removals at times where such observations exist can be used. In this approach, observed disease progress curves are compared to an envelope of progress curves drawn from the predictive distribution. Another approach is to check the event times for the kth individual, as in [26], where the kth removal time is checked various k, where the model that was fitted uses a step function to model the inhomogeneous removal rate.

Several papers have taken the approach of using a $T(y)$ which is a correlation function or a spatial autocorrelation function. For example, in [145], the authors calculate the spatial correlation using a two point correlation function; a modified version of Moran's I statistic for presence and absence data, with a weighting function which is equal to the Euclidean distance between the two points if this distance is in between two specified radii. The spatial autocorrelation was plotted for 100 simulations from each model and compared to the actual observed two point correlation. In [134] a spatial autocorrelation function is used, where a spline correlogram was used as a non-parametric estimator for the autocorrelation function, which was fitted to the observed data.

These test statistics succeed in quantifying the discrepancy in such a way that tests are able to detect model mis-specification if it exists. However, it is important that test statistics must quantifying discrepancy that is relevant. Since disease models are often used in the selecting the control strategy, for example ring culling, it is of prime concern that the model is capable of predicting which individuals are most at risk of being infected next. Using disease progress curves as test statistics can identify model mis-specification regarding the predicted numbers of infections at each time, but it does not directly measure model mis-specification with regards to spatial interactions in the transmission process.

In [182], a paper on model fit for the models for foot-and-mouth disease that were used in [100] used measures of model mis-specification that emphasised accuracy of predictions from these models. Several measures of model fit were formulated which compare the predicted states against the observed states at some future time. The authors use a matrix of the number of hosts in the observed versus predicted states averaged over multiple realisations of the data from the fitted model. This matrix is used in calculating these measures of discrepancy. Whilst the framework for model fitting was frequentist, these methods can be easily adapted to the Bayesian approach of using posterior predictive p-values, with the multiple realisations from the fitted model being replaced by draws of replicate data from the posterior predictive distribution.

From these examples, it can be concluded that posterior predictive checking can be successfully used in determining model fit in the absence of specifying an alternative model whilst targeting aspects of mis-specification which are relevant to the problem, such as determining the control measures, or prior hypotheses regarding model mis-specification. However, as noted in [69], with the use of low dimensional test statistics, there is a risk of oversimplifying what is inherently a complex phenomenon. Whilst easy to interpret, the attempts to reduce the quantification of mis-specification to a single value may reduce the power of posterior predictive tests [69]. Several tests have been developed to increase sensitivity of the tests to mis-specification by using more complex measures to quantify model mis-specification.

3.3.1 Discrepancy measures and posterior predictive checking

These tests use a generalisation of the test statistic by using test statistics which are functions of the parameters, which are called *discrepancy measures* [125, 60]. This idea is analogous to the Z-test of the mean of a normal distribution with known variance σ^2. The test statistic is both a function of the observed data y and the parameter σ^2.

More formally, the posterior predictive p-value of a discrepancy measure $T(x, \theta)$ is expressed as:

$$p(y) = \int P\left(T(x^{rep}, \theta) > T(x, \theta) | x, \theta\right) \pi(x, \theta | y) dx\, d\theta$$

For example, consider the model for Y where y is an i.i.d. sample of size n from $N(\mu, \sigma)$, the discrepancy measure $\frac{\bar{y}-\mu}{\sqrt{\frac{\sigma^2}{n}}}$ would be a logical choice.

3.3.2 Test Statistic Construction Using Latent Residuals Through the Functional Model Representation of Epidemic Models

A method of creating test statistics for epidemic models where there is unobserved data uses what can be called latent residuals. Test statistics can be created for epidemic models by the use of what is known as latent residuals. The concept of latent residuals is based on the ideas of functional-model representations put forward by [41], and the ideas of a generalised residuals put forward by [39], applied to epidemic model checking. Constructing tests based on latent residuals allows the development of tests oriented at detecting specific aspects of mis-specification.

The construction of latent residuals is based on the following reasoning: Consider set of the models $\pi(y, r_1|\theta), \pi(y, r_2|\theta), \pi(y, r_3|\theta) \ldots$ which all share the same marginal model $\pi(y|\theta)$ and prior $\pi(\theta)$, where $r_1, r_2, r_2 \ldots$ are different choices of latent (or unobserved) process. The observed data y do not contain the information about the model adequacy of each model in the set relative to the other models in the set. Thus, evidence against one model in the set is evidence against all models in the set. Thus, the adequacy of the model can be evaluated by choosing a latent process (with marginal model $\pi(y|\theta)$) of known distribution given the fitted model.

Definition 14 (Latent Residual)**.** Consider a function h_θ such that the data (including both the observed and unobserved data) $x = h_\theta(r)$. The function $h_\theta(x)$

can be specified to be any invertible function as desired as long as the marginal model remains the same, and the distribution of r is known under the assumed model. The vector r can be treated as a vector of *latent residuals* and it is used to detect mis-specification of the epidemic model in way analogous to how the residuals from a fitting of a linear model can be used to diagnose mis-specification of a linear model.

Since the distribution of r is known under the assumed model, one can test the model adequacy of model $\pi(y, r|\theta)$, and thus all models in the set $\left\{\pi(y, r|\theta) | \pi(y, r|\theta) \text{ has marginal model } \pi(y|\theta)\right\}$, and thus the model adequacy of $\pi(y|\theta)$:

1. Choose r such that:
   (a) The marginal model of $\pi(y, r|\theta)$ is $\pi(y|\theta)$.
   (b) The distribution of r, $\pi(r)$, is known under the assumed model
2. Choose a discrepancy statistic $T(r)$. $T(r)$ should be a discrepancy measure between r and its known distribution $\pi(r)$.
3. Sample from posterior distribution $\pi(\theta, r|\boldsymbol{y})$.
4. From each sample (θ, r), calculate $p(T, r)$, the latent p-value of the imputed r, using discrepancy measure T (specified earlier), for each sample.
5. The sample mean of the samples of the latent p-value, obtained in the previous step, is a Monte-Carlo estimate of the posterior predictive p-value.

This approach was first used in [66] where the approach was used to assess model fit of an SI model fit to data of a fungal pathogen known as R. Solani in radish. The Sellke thresholds [161] of each individual post were used as residuals to assess model fit of this spatio-temporal model. The consistency of the residuals with an exponential distribution was tested using the Kolmogorov-Smirnov test. That is, given the snapshot data, the infection times of each individual post was imputed, and from this the Sellke thresholds were imputed and used as latent residuals, to

$$\begin{tikzcd}
 & \text{Observation } y = g(x) & \\
 & \text{"Complete" data } x = h(\tilde{\mathbf{r}}, \theta) \arrow[u] & \\
\tilde{r}_j \sim \text{Unif}(0,1) \arrow[ur] & & \theta \sim \pi(\theta) \arrow[ul]
\end{tikzcd}$$

Figure 3.3.2: Diagram of the latent residuals framework

impute the results of a Kolmogorov-Smirnov test. This yielded a distribution of p-values which were used to detect mis-specification of the model.

The paper [111] extended this approach, with a focus on determining the adequacy of the spatial kernel used in each model, with simulated and real-world data. Four sets of latent residuals $\tilde{\mathbf{r}} = (\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4)$ were used, where each $\tilde{r}_j$, $j = 1, 2, 3, 4$ determines a different aspect of the epidemic process. Each $\tilde{r}_j$ is a vector of indeterminate length. Under the assumed model each of these residuals would be a vector of i.i.d. Unif$(0, 1)$ random variables.

The kth element in the vector $\tilde{r}_{1k}$ is the total infectious challenge over all susceptibles at time t_k (population level Sellke threshold) which determines the time of the kth exposure event. The residuals $\tilde{r}_{2k}$ determine the infection link (determines which susceptible became infected due to contact with which infective). $\tilde{r}_{3k}$ gives the quantile of E to I sojourn time. $\tilde{r}_{4k}$ gives the quantile of I to R sojourn time.

This separates the information in the data into various components which could be tested individually (see fig. 3.3.2), allowing the construction of tests targeted at distinct aspects of the epidemic process through the consideration of the processes $\tilde{r}_j$, $j = 1, 2, 3, 4$. Of particular interest are the infection link residuals $\tilde{r}_{2k}$, which determine which I-S pair are responsible for each exposure, which were found to be particularly effective at detecting mis-specification of spatial kernel:

Definition 15 (Infection Link Residual)**.** The infection link residual $\tilde{r}_{2k}$ determines the S-I pair responsible for the kth exposure event according to the following:

Let the kth exposure event be between hosts i and j with probability $p_{ij} \propto \beta K(\mathbf{x}_i, \mathbf{x}_j, \kappa)$. Primary infection is treated as infection from a notional infector with force of infection α.

For all $m \in S(t_k)$ and $n \in I(t_k)$, let $p_{mn} \propto \beta K(\mathbf{x}_m, \mathbf{x}_n, \kappa)$.

Let the p_{mn} be ordered such that $p_{(1)} \leq p_{(2)} \leq p_{(3)} \leq p_{(4)} \leq \ldots$.

Let s' be such that $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf \left\{ s | \tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)} \right\} = s'$$

Hence, the Infection Link Residual can be used to check model adequacy as follows:

1. Choose r to be $\tilde{r}_{2k}$:
    (a) The marginal model of $\pi(y, \tilde{r}_{2k}|\theta)$ is $\pi(y|\theta)$.
    (b) The distribution of r, $\pi(\tilde{r}_{2k})$, is known under the assumed model to be Unif$(0, 1)$
2. Choose a discrepancy statistic $T(r)$. $T(r)$ tests that $\tilde{r}_{2k}$ is a sequence of Unif$(0, 1)$ random variables.
3. Sample from posterior distribution $\pi(\theta, \tilde{r}_{2k}|\boldsymbol{y})$.
    (a) This cannot be performed directly, so instead, since RJMCMC can be used to sample from $\pi(\theta, \boldsymbol{x}|\boldsymbol{y})$, sample from$\pi(\theta, \boldsymbol{x}|\boldsymbol{y})$ and impute $(\theta, \tilde{r}_{2k})$ from $(\theta, \boldsymbol{x})$ (described below).
4. From each sample $(\theta, \tilde{r}_{2k})$, calculate $p(T, \tilde{r}_{2k})$, the latent p-value of the imputed r, using discrepancy measure T (specified earlier), for each sample.
5. The sample mean of the samples of the latent p-value, obtained in the previous step, is a Monte-Carlo estimate of the posterior predictive p-value.

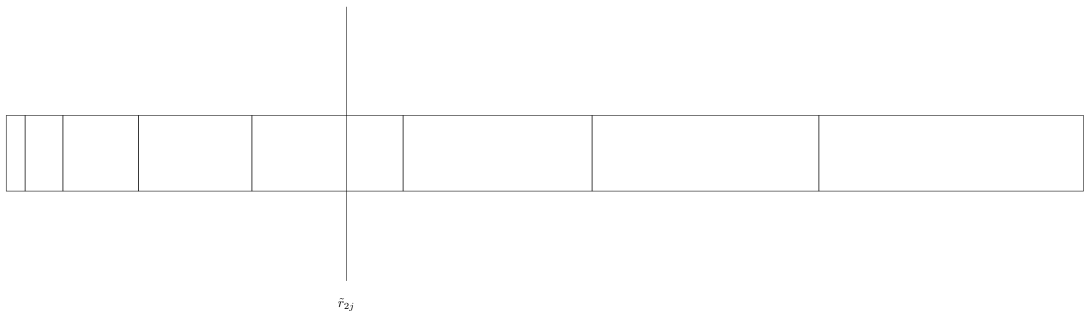

Figure 3.3.3: Diagram of final stage of algorithm to impute the ILR for the kth infected individual

The infection link residual test is embedded within the RJMCMC and imputed using the following algorithm:

1. The infection link for the kth exposure between individuals i and j is chosen with probability p_{ij} from the possible links at time t_k.
2. The infection links are then ordered and the ranking s' of p_{ij} is determined.
3. Generate a random deviate from Unif($\sum_{l=1}^{s'-1} p_{(l)}, \sum_{l=1}^{s'} p_{(l)}$). This is the imputed infection link residual for the kth exposure (fig. 3.3.3).

Since this test will be extended as part of this thesis, it would be valuable to elaborate further as to the rationale behind the reason for ordering the links by size: as described in the supplementary material for the paper [111], the motivation for this choice of the infection link residual (ILR) as a test statistic for detecting mis-specification of spatial kernel can be seen in fig. 3.3.4. Suppose the actual kernel that the data has been generated from is $K_{actual}(\kappa, d)$ and the kernel that has been fitted to the data is $K_{fitted}(\kappa, d)$. Since the actual kernel is longer tailed than would be expected under the fitted model, the infection links will have a tendency to be too small, thus $\tilde{r}_{2k}$ would be non-uniform, and discrepancy from the null hypothesis can be detected by using a test of uniformity, e.g. the Anderson-Darling test for uniformity.

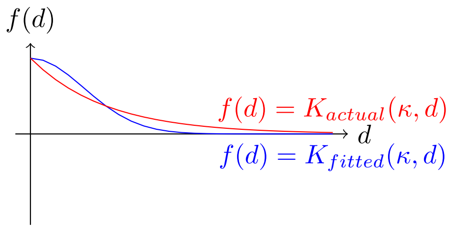

Figure 3.3.4: Diagram of motivation for the infection link residual (ILR) $\tilde{r}_{2k}$

3.3.3 Latent Likelihood-based Tests for Epidemic Models

The posterior predictive testing framework allows the construction of tests which can be used to compare two competing models. This type of tests can be useful in certain circumstances, such as when comparing two models with different spatial kernels (as opposed to assessing the adequacy of a spatial kernel of a model). In addition, the use of alternative model may potentially make the test more able to detect differences from the null hypothesis than a test that has no fixed model to compare against.

A natural way to do this is by embedding the likelihood ratio test within the posterior predictive testing framework. Suppose we are interested in two models M_1 and M_2 with parameter vectors θ and θ' respectively. Let $\pi_1(x|\theta)$ be the likelihood of θ under model M_1, and let $\pi_2(x|\theta')$ be the likelihood of θ' under model M_2. Let $\widehat{\theta}'(x)$ be the maximum likelihood estimate of θ' given x, obtained by maximising $\pi_2(x|\theta')$.

Thus, we can compare $M_1$and M_2 by sampling from $\pi(\theta, \mathbf{x}|y, M_1)$ and calculating from these samples $P\left(T(x^{rep}, \theta) > T(x, \theta)|x, \theta\right) = p(x, T, \theta)$, where $T(x, \theta) = \frac{\pi_1(x|\theta)}{\pi_2(x|\widehat{\theta}'(x))}$. If there is a large amount of posterior support for the p-value being small, this indicates a high amount of discrepancy from the null hypothesis.

Whilst this method compares two models, it has the benefit that it avoids fitting two models to the data, only fitting the first model. The p-value can be obtained by asymptotic approximations if the models are nested. If the models are non-nested the p-value can be calculated by using a nested Monte Carlo algorithm embedded within the data-augmented RJMCMC:

1. Every K RJMCMC iterations, perform the following steps (where i is the

current MCMC iteration, $x^{(i)} = (y, z^{(i)})$ and $\theta^{(i)}$ are the current value of the full data and the parameters θ at MCMC iteration i):

(a) Calculate $T(x^{(i)}, \theta^{(i)}) = \frac{\pi_1(x^{(i)}|\theta^{(i)})}{\pi_2(x^{(i)}|\widehat{\theta'}(x^{(i)}))}$ where $\widehat{\theta'}(x^{(i)})$ is the value of the maximum-likelihood estimate of θ' given $x^{(i)}$

(b) To estimate the p-value via Monte-Carlo, for n times:

   i. Generate a new data set x^*under H_0 given $\boldsymbol{\theta}^{(i)}$

   ii. Calculate $T(x^*, \theta^{(i)}) = \frac{\pi_1(x^*|\theta^{(i)})}{\pi_2(x^*|\widehat{\theta'}(x^*))}$

(c) The estimate of the p-value is $\hat{p} = \frac{\text{Count}\left(T(x^*,\theta^{(i)})>T(x^{(i)},\theta^{(i)})\right)}{n}$

All the models considered in this thesis M_1 and M_2 will share a common latent process. When this is not the case it may nevertheless be possible to compare models by creating a common latent process [67].

Note that this method is essentially the same as a posterior predictive test conditioning on M_1 of the AIC difference between the two models, as the parameter difference between the models is a fixed constant. In contrast with the DIC, since the test is embedded within the posterior predictive framework, there is an unified Bayesian perspective from which the latent AIC of both models is calculated, making the results of such tests easier to interpret. The DICs of two competing models are calculated given each model alone, but there is no DIC calculated conditioning on the other model, so the models are never compared from one single perspective, unlike the latent likelihood ratio test.

In [173] this was the method used to compare models with exponential Sellke thresholds versus models with Weibull Sellke thresholds. This paper demonstrates that the latent likelihood ratio testing method is capable of selecting the correct model, through the use of simulated data to test the effectiveness of this model comparison method.

3.4 Our contributions to model comparison and criticism

In subsequent chapters we will build on the approaches of Section 3.3 by focusing on the following challenges:

1. Likelihood based tests statistics for model comparison (vs. model selection), with an emphasis on spatial kernel assessment.
2. Latent residual based test-statistics for detection of anisotropy in spatial kernels.
3. Massively parallel algorithms for the calculation of the likelihood and ILR test, allowing the model fitting and model assessment techniques detailed in this thesis to be applied to large or complex data-sets.

3.4.1 Likelihood based test statistics for model comparison and detection of anisotropy

Since the latent likelihood ratio test is able to select the correct model in the context of non-spatio-temporal epidemics, the next logical step would be to determine its effectiveness in comparing spatio-temporal models. In such models the spatial kernel is crucial in determining the control measure to be taken against an epidemic, for example determining the culling radius in a ring culling strategy, it is crucial that such test statistics are able to select the correct spatial kernel out of two competing spatial kernels. A test statistic that will be investigated in this thesis is the likelihood ratio test statistic with the full likelihood:

$T(x, \theta) = \frac{\pi_1(x|\theta)}{\pi_2(x|\hat{\theta}(x))}$

where $\pi_1(x|\theta)$ is the likelihood under model 1, $\pi_1(x|\theta)$ is the likelihood under model 2, and $\hat{\theta}(x)$ is the maximum-likelihood estimator of θ.

This thesis aims to evaluate its effectiveness in relation to the infection link residual test statistic described in earlier sections from [111]. Model reinforcement can lead to a loss of power of the test so therefore it may be wise to also consider the likelihood ratio test with the partial likelihood. By limiting the amount of imputed information in the test statistic it may be possible to lower the amount of model reinforcement that occurs. In previous work, it was found that the infection link residuals test, which uses the sets of potential infectors at each infection time, but not the infection times themselves was found to be an effective test of model adequacy. In this vein, the partial likelihood used will be:

$$\begin{aligned} l_{partial}(\boldsymbol{\theta}; \mathbf{t}_E, \mathbf{x}) &= \sum_{\{i | t_E(\mathbf{x}_i) \leq T\}} \log \left[\frac{C(\mathbf{x}_i, t_E^{(i)})}{\sum_{\{j | \mathbf{x}_j \in S(t_E^{(i)})\}} C(\mathbf{x}_j, t_E^{(i)})} \right] \\ \text{where: } C(x,t) &= \alpha + \beta \sum_{y \in I(t)} K(\mathbf{x}, \mathbf{y}, \theta) \\ K(\mathbf{x}, \mathbf{y}, \theta) &= \text{Transmission Kernel} \end{aligned}$$

This partial likelihood can be interpreted as: for each $S \longrightarrow E$ transition i, the likelihood that particular i got infected given the S and I hosts at time of infection. This can also be seen as similar to the Cox partial likelihood for survival models. This partial likelihood was used by [46] for model fitting (but not assessment of model fit). This thesis aims to determine the effectiveness of this test statistic, in addition to that of the full latent likelihood ratio statistic and the infection link residual test statistic.

3.4.2 Latent residual based test-statistics for detection of anisotropy

As stated earlier it is important to focus on aspects of mis-specification which are most pertinent to the purposes that the model will be put. In many situations it is important to test the assumption that the kernel is isotropic. Using the idea of latent residuals we can devise a test statistic for testing for the presence of anisotropy, extending previous work on latent residual tests (see [111]):

Definition 16 (Directional Infection link residuals)**.** Suppose that host j is infected by infectious host i and that this is the kth exposure event. The infection link residual for this infection time is defined by the following:

Let the kth exposure event be between hosts i and j with probability $p_{ij} \propto \beta K(\mathbf{x}_i, \mathbf{x}_j, \kappa)$. Primary infection is treated as infection from a notional infector with force of infection α.

For all $m \in S(t_k)$ and $n \in I(t_k)$, let $p_{mn} \propto \beta K(\mathbf{x}_m, \mathbf{x}_n, \kappa)$.

Order the p_{mn} such that $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the cosine of the angle between the infection link and the vector $(1, 1)$.

Let s' be such that $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf \left\{ s | \tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)} \right\} = s'$$

We also used an alternate version where $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the **angle** between the infection link and the vector $(1, 1)$.

This test statistic is based on the ILR test statistic mentioned above but it has been extended to testing for anisotropy by ordering the links by angle or the cosine of the angle instead of the size of the infection links. The latent likelihood ratio test statistics detailed above can also be used to test for anisotropy, as long as the form of the anisotropic kernel to be tested against is known, and therefore can only be used to compare models with specific anisotropy. In addition, the latent likelihood ratio test can be used to test between different anisotropic kernels.

Chapter 4

Latent Likelihood Tests for Epidemic Models

4.1 Introduction

In this chapter, the ability of the infection link residuals (ILR) and latent likelihood ratio tests (LLR) to detect mis-specification of spatial transmission kernel will be compared using simulated data. Both forms of the LLR tests will be used: one using the full likelihood, and one using a partial likelihood which only uses information about the order of infections (not the actual infection times). The analysis performed in this chapter consists of initial exploratory runs to determine general trends, and then further runs to verify these trends hold over a variety of datasets. The datasets that will be used are simulated data, which are generated with known parameters, model and spatial kernel. In order to compare the ability of the aforementioned tests to detect discrepancy between the fitted model and data, the model fitted to the data will use a spatial kernel different to the kernel used in the model that the data was generated from.

4.2 Methodology

4.2.1 Generation of Simulated Data

The data were generated using the Gillespie-based algorithm described in Section 2.2. The model used to generate the data is the model described in Section 2.1 on page 11, an SEIR model consisting of states: Susceptible S, Exposed E, infectious I and Removed R. Members of the population transition from S to E to I to R, and the transition is only in one direction, that is, there are no transitions in the reverse direction. Hosts in state I can infect hosts in state S which then transition to state E upon infection. When hosts transition to state R they cannot transition any further and are no longer infectious.

The force of infection is given by:

$$C(x,t) \quad = \quad \alpha + \beta \sum_{y \in I(t)} K(\boldsymbol{x}, \boldsymbol{y}, \kappa)$$

where $K(\boldsymbol{x}, \boldsymbol{y}, \kappa)$ is the transmission kernel, α is the primary infection rate, and β is the secondary infection rate.

The distributions of the waiting times for states E and I are gamma distributions, parametrised by their means μ_E, μ_I and variances σ_E^2, σ_I^2. In a setting analogous to that in the paper [111], the transition times into state E are not observed but transition times into state I and R during the time interval $[0, T]$ are observed. The hosts are uniformly distributed over a square region.

4.2.2 Likelihood

As in 2.2.5, in Section 2.2.2 on page 14 the likelihood can be expressed in the form:

$$\begin{aligned} L(\alpha,\beta,\kappa,\mu_E,\sigma_E^2,\mu_I,\sigma_I^2|\mathbf{x}) \quad \propto \quad & \left(\prod_{i;t_E^{(i)}\leq T} \exp\left[-\int_0^{t_E^{(i)}} C(x_i,t)\,dt\right]\cdot C(x_i,t_E^{(i)})\right)\cdot \\ & \left(\prod_{i;t_E^{(i)}>T} \exp\left[-\int_0^{T} C(x_i,t)\,dt\right]\right)\cdot\left(\prod_{i;t_I^{(i)}\leq T} f_E(t_I^{(i)}-t_E^{(i)};\mu_E,\sigma_E)\right)\cdot \\ & \left(\prod_{i;t_I^{(i)}>T>t_E^{(i)}} \left(1-F_E(T-t_E^{(i)};\mu_E,\sigma_E)\right)\right)\cdot\left(\prod_{i;t_R^{(i)}\leq T} f_I(t_R^{(i)}-t_I^{(i)};\mu_I,\sigma_I^2)\right)\cdot \\ & \left(\prod_{i;t_R^{(i)}>T>t_I^{(i)}} \left(1-F_I(T-t_I^{(i)};\mu_I,\sigma_I^2)\right)\right). \end{aligned}$$

The likelihood function, which is key to the iterative calculation of all the estimates in this chapter is often extremely computationally intensive to calculate. Algorithms have been derived in this thesis to parallelise these calculations in such a way that the graphics processor (which is normally used for matrix calculations for the real-time generation of 3D graphics) of the computer can be used to accelerate the calculation process. These algorithms will be detailed later in the thesis.

4.2.3 Prior Specification

A Unif$(0, M)$ uniform prior was used for $\alpha, \mu_E, \sigma_E^2, \mu_E, \sigma_E^2$, where $M \approx 1.7 \times 10^{308}$ is the computer limit for double precision floating point numbers in C++.

The prior distributions used for the other parameters were:

$$\beta \sim \Gamma(\mu = 1, \sigma^2 = 100)$$
$$\kappa \sim \Gamma(\mu = 1, \sigma^2 = 100)$$

4.2.4 Calculation of the Posterior Expected Imputed P-Value

The process for the computation of the expected posterior imputed p-value is embedded within the data augmented MCMC (a special case of RJMCMC, and is sometimes referred to herein as RJMCMC) that was described in Section 2.3.3 on page 34.

4.2.4.1 Data Augmented MCMC

Recall that in Section 2.3.3, the algorithm used for data augmented MCMC (sometimes referred as RJMCMC herein; DAMCMC is a special case of RJMCMC) is:

For each iteration, the following process is repeated:

1. Let the current state of the algorithm at time $k = 0, 1, 2, \ldots$ be denoted by $(\boldsymbol{\theta}^{(k)}, \mathbf{z}^{(k)})$ where $\boldsymbol{\theta}^{(k)} = (\alpha^{(k)}, \beta^{(k)}, \kappa^{(k)}, \mu_E^{(k)}, \sigma_E^{2(k)}, \mu_I^{(k)}, \sigma_I^{2(k)})$ and the times of transition into the exposed state be denoted by $\mathbf{z}^{(k)} = z_1^{(k)}, z_2^{(k)}, \ldots, z_N^{(k)}$ where N is the number of hosts where N is fixed (and hence the length of $z^{(k)}$ is fixed).

2. Using the Metropolis-Hastings algorithm, update each of the parameters in the parameter vector individually. That is, for each parameter $\phi^{(k)}$ in $\boldsymbol{\theta}^{(k)}$:

   (a) Draw a proposal value ϕ' from the proposal distribution $q(\phi'|\phi^{(k)})$.

   (b) Calculate the probability of acceptance α. Let $\boldsymbol{\theta}^*$ be $\boldsymbol{\theta}_t$ with $\phi^{(k)}$ replaced by ϕ':

   $$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^*|\mathbf{z}^{(k)}, \mathbf{y}) \cdot q(\phi'|\phi^{(k)})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y}) \cdot q(\phi^{(k)}|\phi')}\right)$$

   (c) With probability α, set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^*$, otherwise set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)}$.

3. For each data item $z_i^{(k)}$ in $\mathbf{z}^{(k)}$:

   (a) **if** $t_I^{(i)} < T$ set move type to *Standard*.

   (b) **else :**

      i. **if** $z_i^{(k)}$ does not fall within $[0, T]$ set move type to *Addition*

      ii. **else** set move type to *Shift* or *Deletion* with probability $\frac{1}{2}$

   (c) **if** move type is *Standard, Addition* or *Shift*: Generate proposal $z_i^{(k)*} \sim \text{Unif}(0, T)$

(d) **else** set proposed value of $z_i^{(k)*}$ to be outside $[0, T]$

(e) **if** move type is *Standard* or *Shift* set $\nu = 1$

(f) **else if** move type is *Addition* $\nu = \frac{T}{2}$

(g) **else if** move type is *Deletion* set $\nu = \frac{2}{T}$

(h) Set $\mathbf{z}^* = z_1^{(k)}, \ldots, z_i^{(k)*}, \ldots, z_N^{(k)}$. Then the acceptance probability α is given by:

$$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^*, \mathbf{y})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y})} \cdot \nu\right)$$

(i) With probability α, set $\mathbf{z}^{(k+1)} = \mathbf{z}^*$ otherwise $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}$

The above algorithm is modified to make it more efficient, by using an independence sampler for the proposal distributions for z_i for the cases where $I_i < T$. Thus, in this case the proposal distribution and acceptance ratio (for the *Standard* moves in Step 3 above) is [140, 107]:

$$q(I_i - z_i, I_i - z_i^*) \equiv \text{Gamma}(\mu_I, \sigma_I^2)$$
$$\alpha = \min\left(1, \frac{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^*, \mathbf{y})}{\pi(\boldsymbol{\theta}^{(k)}|\mathbf{z}^{(k)}, \mathbf{y})} \cdot \frac{q(\mathbf{z}^*|\mathbf{z}^{(k)})}{q(\mathbf{z}^{(k)}|\mathbf{z}^*)}\right)$$

for the cases where it is known that $I_i < T$.

4.2.4.2 Embedding the Tests within DAMCMC

To embed the tests within the data augmented MCMC, the following steps were added to the algorithm above as an extra step:

4. If $k \bmod K = 0$ where K is a positive integral value chosen by the user, calculate the test statistic(s).

5. From each test statistic, calculate its p-value and store the p-value obtained.

The following sections describe the calculation of the test statistics and their relevant p-values based upon the full data x which includes both the observed data y and imputed data z.

4.2.4.3 Infection Link Residuals (ILR) Test Statistic and Imputed P-Value Calculation

4.2.4.3.1 Calculation of Test Statistic As introduced in Section 3.3.2 on page 73, and the paper [111], recall that the definition of the infection link residuals (ILR) test is:

Definition 17 (Infection Link Residual)**.** The infection link residual $\tilde{r}_{2k}$ determines the S-I pair responsible for the kth exposure event according to the following:

Let the kth exposure event be between hosts i and j with probability $p_{ij} \propto \beta K(\mathbf{x}_i, \mathbf{x}_j, \kappa)$. Primary infection is treated as infection from a notional infector with force of infection α.

For all $m \in S(t_k)$ and $n \in I(t_k)$, let $p_{mn} \propto \beta K(\mathbf{x}_m, \mathbf{x}_n, \kappa)$.

Let the p_{mn} be ordered such that $p_{(1)} \leq p_{(2)} \leq p_{(3)} \leq p_{(4)} \leq \dots$.

Let s' be such that $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf\left\{s | \tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)}\right\} = s'$$

The infection link residual test is embedded with the RJMCMC and calculated by the following algorithm (as mentioned in Section 3.3.2 on page 75, from [111]):

1. The infection link for the kth exposure between individuals i and j is chosen with probability p_{ij} from the possible links at time t_k. Primary infection is treated as being an infection caused by a notional infector with force of infection α.

2. The infection links are then ordered and the ranking s' of p_{ij} is determined.

3. Generate a random deviate from Unif($\sum_{l=1}^{s'-1} p_{(l)}, \sum_{l=1}^{s'} p_{(l)}$). This is the imputed infection link residual for the kth exposure.

4.2.4.3.2 Calculation of Imputed P-Value The p-value is calculated using the Anderson-Darling test [7]. This is a frequentist test of the hypotheses:

$$H_0 \text{ :The data has cumulative distribution function } F(x)$$

$$H_A \text{ :The data does not have cumulative distribution function } F(x)$$

The data for this test is a random sample denoted $\{X_1, X_2, \ldots, X_n\}$

Let the empirical distribution function be defined as:

$$F_n(x) = \frac{\text{number of } X_1, X_2 \ldots, X_n \text{ that are} \leq x}{n}$$

The test statistic is defined as:

$$A_n = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\ln F(X_i) + ln(1 - F(X_{n+1-i}))\right] \tag{4.2.1}$$

The Anderson-Darling test statistic can be expressed in another form, which shows that it is the integral of the weighted squared difference between the empirical distribution function and the hypothesised distribution function, multiplied by a weighting with weight concentrated towards the tails of the distribution.

$$A_n = n\int_0^1 \frac{[F_n(x) - F(x)]^2}{F(x)(1-F(x))}dF(x) \tag{4.2.2}$$

This makes the Anderson-Darling test more able to detect discrepancy between the hypothesised distribution and the data and the tails of the distribution than the Kolmogorov-Smirnov test which is more commonly used.

To obtain the test statistic in Equation 4.2.1, use partial fractions on Equation 4.2.2:

$$A_n = n\left(\int_0^1 \frac{[F_n(x) - F(x)]^2}{F(x)} dF(x) + \int_0^1 \frac{[F_n(x) - F(x)]^2}{(1 - F(x))} dF(x)\right) \tag{4.2.3}$$

Since the empirical CDF $F_n(x)$ is a step function, it is straightforward to obtain the test statistic given in Equation 4.2.1.

In this case $F(x)$ is a uniform cdf between 0 and 1, the hypotheses and test statistics simplify to:

H_0 :The data has cumulative distribution function x

H_A :The data does not have cumulative distribution function x

The test statistic is simplified to:

$$A_n = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\ln X_i + ln(1 - X_{n+1-i})\right] \tag{4.2.4}$$

Regarding the derivation of the test statistic, start with:

$$A_n = n\int_0^1 \frac{[F_n(x) - x]^2}{x(1-x)} dx$$

Use partial fractions on the integral to obtain:

$$A_n = n\left(\int_0^1 \frac{[F_n(x) - x]^2}{x} dx + \int_0^1 \frac{[F_n(x) - x]^2}{(1-x)} dx\right)$$

Since the empirical distribution function $F_n(x)$ of the data is a step function, it is straightforward to integrate and simplify to obtain the test statistic in Equation 4.2.4.

The Anderson-Darling test is performed upon the infection link residuals that are obtained through the algorithm described on page 86.

4.2.4.4 Latent Likelihood Ratio Test (LLR) Test Statistic and Imputed P-Value Calculation

4.2.4.4.1 Full Likelihood LLRT Recall that the test statistic used for the full likelihood latent likelihood residual test (LLRT), given in Section 3.4.1 on page 78, is:

$$T(x,\theta) = \frac{L_1(\theta;x)}{L_2(\hat{\theta}(x);x)}$$

where $L_1(\theta;x)$ is the likelihood under model 1, $L_2(\theta;x)$ is the likelihood under model 2, and $\hat{\theta}(x)$ is the maximum-likelihood estimator of θ.

4.2.4.4.2 Partial likelihood LLRT Recall that the test statistic used for the partial-likelihood latent likelihood residual test (Partial LLRT), given in Section 3.4.1 on page 78, is defined as the following:

$$\begin{aligned}
L_{partial}(\boldsymbol{\theta};\mathbf{x}) &= \prod_{\{i|t_E(\mathbf{x}_i)\le T\}} \frac{C(\mathbf{x}_i, t_E^{(i)})}{\sum_{\{j|\mathbf{x}_j\in S(t_E^{(i)})\}} C(\mathbf{x}_j, t_E^{(i)})} \\
\text{where: } C(x,t) &= \alpha + \beta \sum_{y\in I(t)} K(\mathbf{x},\mathbf{y},\theta) \\
K(\mathbf{x},\mathbf{y},\theta) &= \text{Transmission Kernel}
\end{aligned}$$

$$T_{partial}(x,\theta) = \frac{L_{1,partial}(\theta;x)}{L_{2,partial}(\hat{\theta}(x);x)}$$

where $L_{1,partial}(\theta;x)$ is the likelihood under model 1, $L_{2,partial}(\theta;x)$ is the likelihood under model 2, and $\hat{\theta}(x)$ is the maximum-likelihood estimator of θ.

4.2.4.4.3 Calculation of the Imputed P-Value Without loss of generality, suppose the test statistic used is $T(x,\theta)$ (the same method is used for $T_{partial}(x,\theta)$, except $T(x,\theta)$ is replaced in the following steps with $T_{partial}(x,\theta)$).

1. Calculate $T(x,\theta)$
2. For n_{test} times

   (a) Generate new data-set $\mathbf{x}^*$ under the fitted model given $\boldsymbol{\theta}^{(k)}$

   (b) Evaluate $T'(\boldsymbol{\theta}^{(k)}; \mathbf{x}^*)$

3. Store $\hat{p} = \frac{\text{Count}(\text{T}'(\boldsymbol{\theta}^{(\text{k})};\mathbf{x}^*)>\text{T}(\boldsymbol{\theta}^{(\text{k})};,\mathbf{x}))}{n_{test}}$

In practice, it is only practical to have n_{test} set to 1, as this process takes a large amount of time (data-regeneration, and maximum-likelihood estimation take a relatively large amount of time). Under different values of n_{test}, many different distributions of $\hat{p}$ will be obtained, however, the average of overall MCMC samples of $\hat{p}$ will converge to the expectation of posterior latent p-value regardless of the value of n_{test}. In this thesis, this expected value is used to summarise the whole the distribution because of computing power constraints.

4.3 Exploratory runs

4.3.1 Methodology

Computer runs were performed to investigate the ability of the model comparison methods (detailed above) to detect the mis-specification of spatial kernel. See table 4.2 for results. The data were generated from an exponential kernel with the parameters in Table 4.1 and the Reversible Jump MCMC algorithm was run on this simulated data. This is to test the sensitivity of the model comparison methods when a mis-specification of the kernel is present. The test's ability to detect discrepancy between the fitted model and the data with different amounts of data was performed by truncating the data at different end times T such that the proportion of infectious individuals was a given percentage.

The model that was fitted to the data had a different kernel: the kernels fitted to the simulated data were the Gaussian kernel,

$$\exp\left\{-\kappa d^2\right\}$$

Cauchy kernel

$$(1 + d/\kappa)^{-1}$$

Parameter	Value
α	0.001
β	3.000
κ	0.030
μ_E	5.000
σ_E^2	2.500
μ_I	1.772
σ_I^2	0.858

Table 4.1: Parameter values used for data generation in the exploratory runs.

and power law kernel

$$(1 + d^{\kappa})^{-1}$$

Since one of the most important aspects of a spatial kernel is its tail length, the Gaussian kernel, Cauchy kernel, and power law kernel have been chosen as they have different tail lengths. These kernels were chosen to be fit to the simulated data as they allow the comparison of how the sensitivity of each of the tests is affected by the different tail length of the fitted kernel, versus the tail length of the actual kernel. The Gaussian kernel is similar in tail-length to that of the actual kernel that the data is generated from, the exponential kernel (the Gaussian kernel is exponentially bounded). The power law and Cauchy kernel are quite different from the exponential kernel in tail-length. Fitting these kernels to the data allows us to see the relative sensitivity of each test with regards to how different the spatial kernel fitted to data is versus the actual kernel that the data was generated from.

RJMCMC was used to obtain estimates of the posterior distributions. A burn-in of 1 million parameter iterations was used. The runs took between 6 or seven hours to complete on a HP Z420 workstation with a NVIDIA GTX Titan GPU. The test statistic and p-value were estimated at an interval of 25,500 iterations (of which 10,500 were parameter updates, 15,000 were updates of the augmented data) starting after 1.5 million parameter updates. The RJMCMC was run for approximately a further 10 million parameter updates (approximately 25,000,000 iterations), and not less than 5 million parameter updates. Readers should note that the majority of the time was spent on the calculation of the test statistic

Hypotheses Tested		Simulated data-set	Infection Link Residuals		LLR(full)		LLR(partial)	
H_0	H_A (for LLR tests)	Total % Population Infected	Iterations	$E(\hat{p})$	Iterations	$E(\hat{p})$	Iterations	$E(\hat{p})$
$\exp\{-\kappa d^2\}$	$\exp\{-\kappa d\}$	100	615	0.489	629	0.013	629	0.005
$\exp\{-\kappa d^2\}$	$\exp\{-\kappa d\}$	75	1680	0.500	1680	0.020	1680	0.016
$\exp\{-\kappa d^2\}$	$\exp\{-\kappa d\}$	50	2788	0.504	2073	0.050	2787	0.088
$\exp\{-\kappa d^2\}$	$\exp\{-\kappa d\}$	5	2091	0.503	2091	0.452	2091	0.510
$(1 + d/\kappa)^{-1}$	$\exp\{-\kappa d\}$	40	1347	0.0000012	1346	0.004	1323	0.001
$(1 + d^{\kappa})^{-1}$	$\exp\{-\kappa d\}$	40	894	0.0000445	894	0.056	1099	0.708

Table 4.2: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: data-set, null hypothesis tested and estimated expected p-values from the infection link residuals test, LLR (full likelihood) and LLR (partial likelihood). The "iterations" referred to here, refers to the number of times that the p-value was estimated, which is once every 25,500 Gibbs updates.

and p-value. Trace and density plots for an example run can be found for the runs in Figures 4.3.1 and 4.3.2. Monte Carlo estimates of the posterior parameter means and variances for an example run can be found in figure 4.3.3. High autocorrelation was observed in the chains obtained. Hence, the chains were run for a high numbers of iterations. There appears to be some correlation between the parameters α and β. One way of reducing this in the MCMC is by normalising transmission kernels (which would reduce the correlation between α and β, but this has the disadvantage of making the parameters obtained difficult to interpret). It was decided to keep the parameters un-normalised for this reason.

4.3.2 Results

If the fitted kernel is similar to the actual kernel the LLR tests are more able to detect mis-specification of the spatial kernel than the ILR tests. From Table 4.2, the high values of expected posterior p-value for the ILR test show that the ILR tests failed to find substantial discrepancy where the fitted model used a Gaussian kernel when the actual kernel used to generate the data was an exponential kernel. In contrast, the LLR tests were able to detect substantial discrepancy in the fitted model of a Gaussian kernel (where the actual model which used an exponential kernel), as demonstrated by the low obtained posterior expected p-values in Table 4.2. The ability of the LLR tests to detect this misfit decreases as the amount of observed symptomatic infection data decreases with the full likelihood variant of the LLR outperforming the partial likelihood variant of the LLR when the amount of observed symptomatic infection data decreases, shown by the relative increase in the posterior expected p-values in Table 4.2.

If the fitted kernel is very different from the actual kernel the ILR test appears to be more able to detect mis-specification of spatial kernel then the LLR tests. When a power law kernel or the long-tailed kernel $(1 + d/\kappa)^{-1}$ was fitted to the data (when the simulated data was generated with an exponential kernel), very low expected posterior p-values for the ILR were obtained. These posterior expected p-values were smaller than those obtained for the LLR tests, although the full likelihood LLR tests did find substantial levels of discrepancy, with low posterior expected p-values, although not as low as those obtained from the ILR test. The partial LLR was able to detect discrepancy for the fitted kernel $(1 + d/\kappa)^{-1}$ (where the actual kernel was exponential), with a low expected posterior p-value being obtained despite only 40% of the observed symptomatic infection data being observed. It is found in Table 4.2 that the partial LLR is unable detect discrepancy from the fitted kernel of a power law kernel (when the actual kernel is an exponential kernel) and only 40% of symptomatic infections are observed, with a high posterior p-values being obtained (the chain of obtained test statistics was checked visually and there appeared to be no signs of non-convergence of the MCMC to the stationary distribution nor any obvious problems in the optimiser in finding maxima). This is found to repeat itself in the verification runs in later sections. This shows that there may be a certain degree of difference between actual and fitted kernels where the full Likelihood LLR outperforms the partial likelihood LLR.

4.4 Verification Runs

Having performed exploratory runs on simulated data-sets to identify possible trends, further runs will be performed to verify that these trends re-occur over a wide range of simulated data-sets.

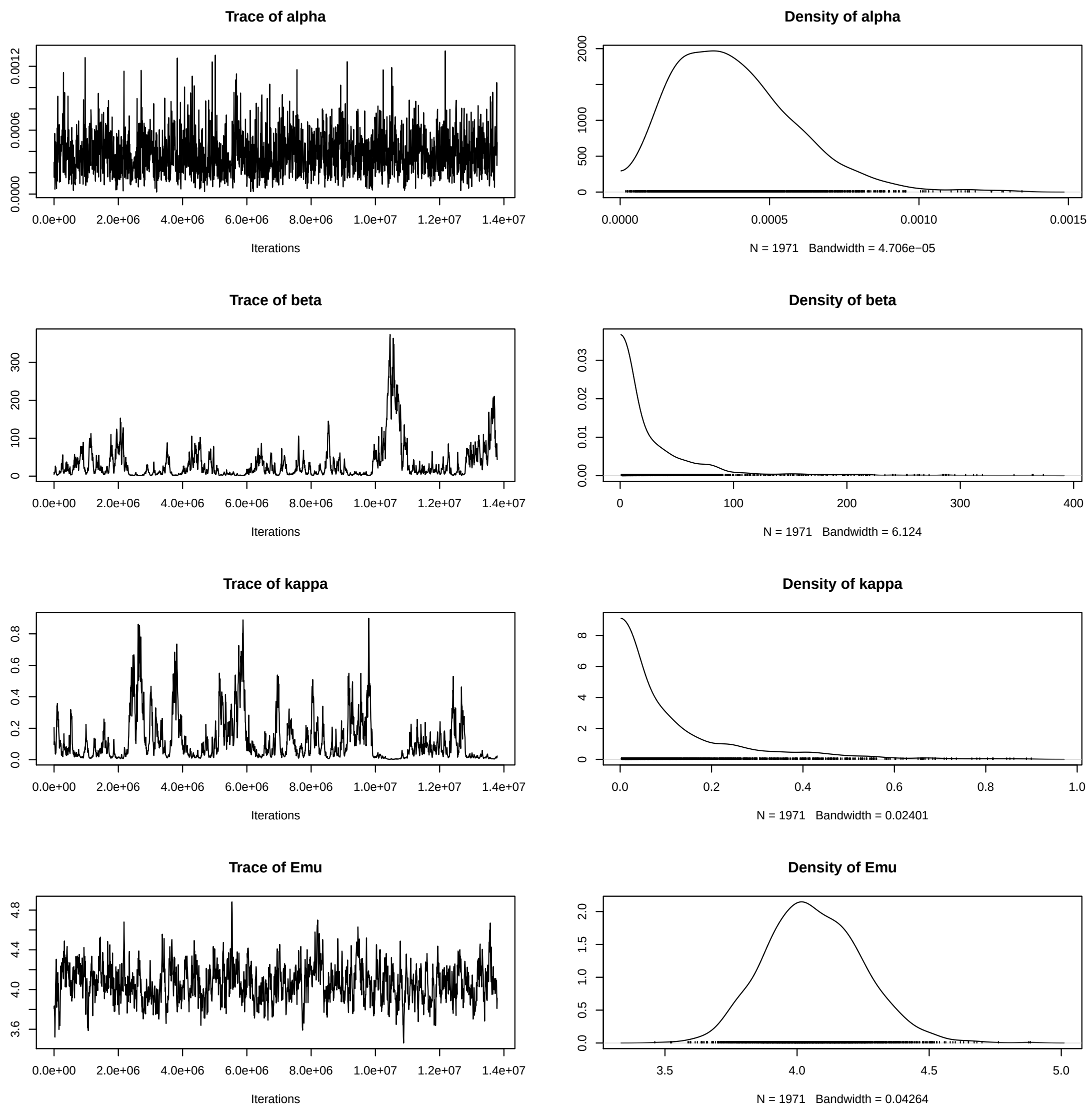

Figure 4.3.1: Trace and density plots of MCMC output for fitting model $H_0 : K(\mathbf{x}, \mathbf{y}, \kappa) = (1 + \frac{|\mathbf{x}-\mathbf{y}|}{\kappa})^{-1}$ vs. $H_A : K(\mathbf{x}, \mathbf{y}, \kappa) = \exp\{-\kappa|\mathbf{x} - \mathbf{y}|\}$. α, β, κ are referred to as "alpha", "beta" and "kappa" in the plots respectively. "Emu" and "Es2" refer to μ_E and σ_E^2.

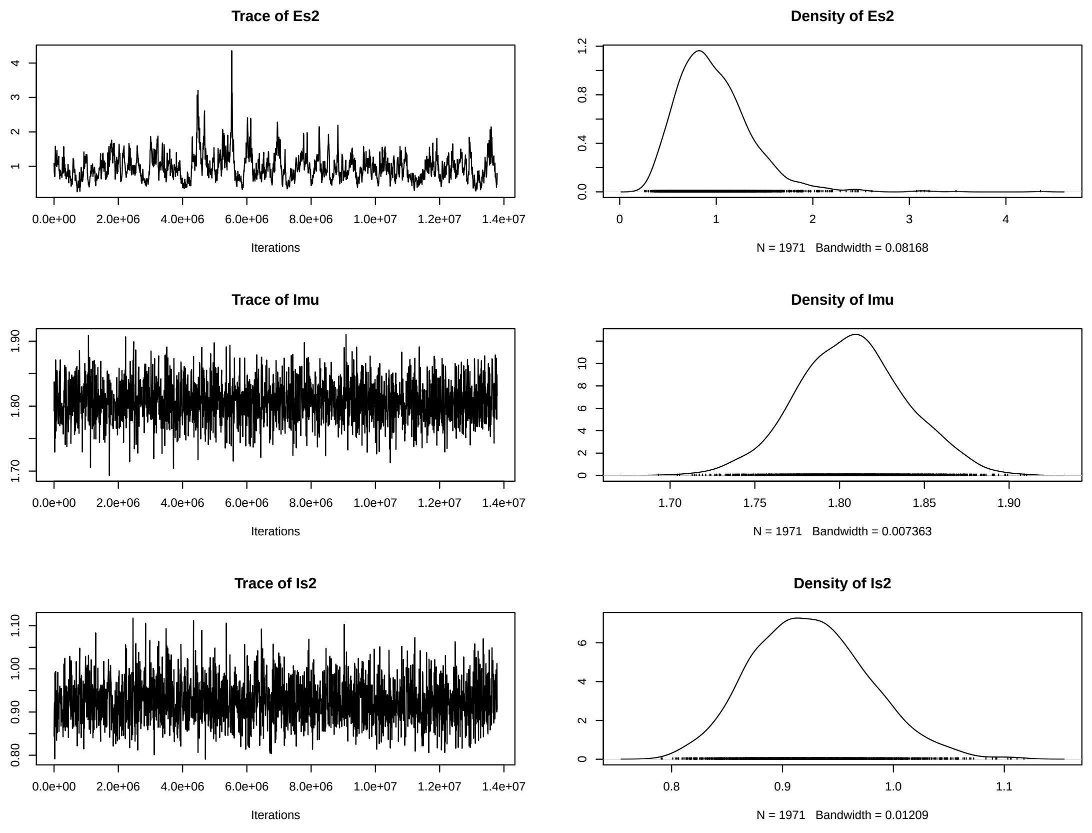

Figure 4.3.2: Trace and density plots of MCMC output for fitting model H_0 : $K(\mathbf{x}, \mathbf{y}, \kappa) = (1 + \frac{|\mathbf{x}-\mathbf{y}|}{\kappa})^{-1}$ vs. H_A : $K(\mathbf{x}, \mathbf{y}, \kappa) = \exp\{-\kappa|\mathbf{x} - \mathbf{y}|\}$. "Emu" and "Es2" refer to μ_E and σ_E^2. "Imu" and "Is2" refer to μ_I and σ_I^2.

```
Iterations = 1:13796987
Thinning interval = 7
Number of chains = 1
Sample size per chain = 1970999

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean        SD  Naive SE Time-series SE
alpha 3.785e-04 2.011e-04 1.432e-07      8.154e-07
beta  3.336e+01 4.762e+01 3.392e-02      7.323e+00
kappa 1.242e-01 1.479e-01 1.054e-04      1.720e-02
Emu   4.066e+00 1.836e-01 1.308e-04      1.038e-02
Es2   9.762e-01 3.892e-01 2.772e-04      2.931e-02
Imu   1.807e+00 3.264e-02 2.325e-05      9.292e-05
Is2   9.254e-01 5.452e-02 3.883e-05      1.554e-04

2. Quantiles for each variable:

           2.5%       25%       50%       75%     97.5%
alpha 8.541e-05 0.0002299 3.463e-04 4.919e-04 8.564e-04
beta  1.864e+00 6.2224700 1.528e+01 4.146e+01 1.868e+02
kappa 5.339e-03 0.0242465 6.594e-02 1.624e-01 5.413e-01
Emu   3.734e+00 3.9400200 4.057e+00 4.185e+00 4.444e+00
Es2   4.170e-01 0.7048780 9.191e-01 1.176e+00 1.864e+00
Imu   1.744e+00 1.7845200 1.806e+00 1.828e+00 1.872e+00
Is2   8.247e-01 0.8875950 9.232e-01 9.608e-01 1.038e+00
```

Figure 4.3.3: Summary of MCMC output for fitting model $H_0 : \; K(\mathbf{x},\mathbf{y},\kappa) = (1 + \frac{|\mathbf{x}-\mathbf{y}|}{\kappa})^{-1}$ vs. $H_A : \; K(\mathbf{x},\mathbf{y},\kappa) = \exp\{-\kappa|\mathbf{x}-\mathbf{y}|\}.\alpha, \beta, \kappa$ are referred to as "alpha", "beta" and "kappa" in the output respectively. "Emu" and "Es2" refer to μ_E and σ_E^2. "Imu" and "Is2" refer to μ_I and σ_I^2.

4.4.1 Methods

The data was simulated from the Gillespie algorithm with the following known parameters:

Parameter	Data-set			
	Original	$\alpha \times 2$	$\beta \times 2$	$\kappa \times 2$
α	0.001	**0.002**	0.001	0.001
β	3.000	3.000	**6.000**	3.000
κ	0.030	0.030	0.030	**0.060**
μ_E	5.000	5.000	5.000	5.000
σ_E^2	2.500	2.500	2.500	2.500
μ_I	1.772	1.772	1.772	1.772
σ_I^2	0.858	0.858	0.858	0.858

The parameters used to generate the simulated data was taken as a starting point. Three different data-sets were generated: $\alpha \times 2$, $\beta \times 2$, $\kappa \times 2$ with the primary infection parameter, secondary infection parameter and kernel parameters doubled from the "Original" parameters respectively.

An exponential kernel was used to generate the data:

$$K(\mathbf{x}, \mathbf{y}, \kappa) \quad = \quad \exp\{-\kappa|\mathbf{x} - \mathbf{y}|\}$$

To determine whether the model testing methods were able to detect a mis-specification of transmission kernel, alternative kernels were fitted to the data, and the model testing methods were used to perform a test of whether the kernel fits the data, or in the case of the likelihood ratio tests, that the kernels were an exponential kernel. There were two alternate kernels used in this case which were the Gaussian kernel

$$\exp\left\{-\kappa d^2\right\}$$

and the power law kernel

$$(1 + d^\kappa)^{-1}$$

The Gaussian kernel is not as long-tailed as the power law kernel, and thus it would be of interest to examine whether the model testing methods react differently when the kernel is mis-specified as a longer tailed kernel or a shorter tailed kernel. In the preliminary runs, a Cauchy kernel was also fitted to the data. However, it was found that it was difficult to tune MCMC to gain adequate mixing with models with this kernel, and the maximum-likelihood estimation algorithms often were difficult to tune to obtain convergence to the maximum likelihood estimate. In addition, the Cauchy kernel, with its very heavy tail, would probably be an unrealistic choice of kernel to fit to the data. It is for these reasons that the Cauchy kernel is not fitted as a model in the set of runs.

It is also of interest to test how the model comparison methods perform when there is a limited amount of data available. Several realisations of the same data-set were used in which the observation period was varied such that observation was stopped at times where certain percentages of the population were infected. These levels used in this thesis were: 40%, 70%, and 100%.

Since the generation of the simulated data requires the random number generator to be started with a seed value, data-sets were also simulated using a different random number seed to generate different realisations of the epidemic with the same parameters as above. This is to verify that the same trends are observable with epidemics which have been generated using different sequences of random numbers.

4.4.1.1 Algorithm

The RJMCMC algorithm from subsection 2.3.3 was used to obtain estimates from the posterior distribution of the parameters and augmented data. The embedded tests were performed every 10,500 iterations, to obtain samples which are relatively independent of each other. In addition, performing the tests was found to require a lot of computational resource, so performing the tests every RJMCMC iteration would be time consuming, in addition to the fact that the RJMCMC samples of the parameters tended to display high amounts of autocorrelation in exploratory runs. The RJMCMC algorithm was tuned to have an acceptance rate of approximately 15 to 20% for the parameter updates. The updates of the augmented data, that is, the unobserved exposure times, were updated through the use of the independence sampler (described in Chapter 2) if $t_I^{(i)} > T$. Otherwise, a uniform proposal was used. Normal distributions were used as proposal distributions for the parameter updates, apart from the parameter updates for β and κ, where generalised t_1 distributions were used instead, since the posterior distributions for these parameters tended to long-tailed and using a more longer-tailed proposal distribution allowed the chain to explore the posterior distribution better. The time to complete each run was approximately seven hours, which mostly consisted of the time spent on the embedded tests. The computer used for the runs in this thesis was a consumer grade gaming computer with a NVIDIA GTX Titan graphics card, which is a consumer grade graphics card. Each run consisted of approximately 10 million Gibbs-within-Metropolis parameter updates, in which a 1% random scan was used to update the unobserved exposure times. Chain mixing generally tends to improve as the percentage of the augmented data updated in each sweep increases, although there is diminishing returns in

data-set	H_0	Total % Population Infectious	Test Iterations	ILR $E(\hat{p})$	LLR (Full) $E(\hat{p})$	LLR (Partial) $E(\hat{p})$
$\alpha \times 2$	$(1+d^{\kappa})^{-1}$	100	2428	0.0000243	0.005319	0.0000000
$\alpha \times 2$	$(1+d^{\kappa})^{-1}$	70	952	0.0002571	0.02473	0.2269000
$\alpha \times 2$	$(1+d^{\kappa})^{-1}$	40	1465	0.0042040	0.1242	0.8014000
$\alpha \times 2$	$\exp\{-\kappa d^2\}$	100	1818	0.4966585	0.0006974	0.0038500
$\alpha \times 2$	$\exp\{-\kappa d^2\}$	70	1431	0.4929907	0.006932	0.0461200
$\alpha \times 2$	$\exp\{-\kappa d^2\}$	40	457	0.4937340	0.06909	0.2801000
$\beta \times 2$	$(1+d^{\kappa})^{-1}$	100	1704	0.0000006	0.0000	0.0000000
$\beta \times 2$	$(1+d^{\kappa})^{-1}$	70	1316	0.0000013	0.01566	0.1771000
$\beta \times 2$	$(1+d^{\kappa})^{-1}$	40	1710	0.0001413	0.1031	0.6801000
$\beta \times 2$	$\exp\{-\kappa d^2\}$	100	1397	0.4963660	0.02189	0.0157500
$\beta \times 2$	$\exp\{-\kappa d^2\}$	70	1196	0.4905312	0.03135	0.0393000
$\beta \times 2$	$\exp\{-\kappa d^2\}$	40	1490	0.4907014	0.1000	0.1295000
$\kappa \times 2$	$(1+d^{\kappa})^{-1}$	100	1545	0.0004014	0.0000000	0.0000000
$\kappa \times 2$	$(1+d^{\kappa})^{-1}$	70	1648	0.0002682	0.0000000	0.0000000
$\kappa \times 2$	$(1+d^{\kappa})^{-1}$	40	1699	0.0000569	0.0000000	0.6845000
$\kappa \times 2$	$\exp\{-\kappa d^2\}$	100	3480	0.4970400	0.0000000	0.0000000
$\kappa \times 2$	$\exp\{-\kappa d^2\}$	70	2806	0.4920980	0.0000000	0.0021380
$\kappa \times 2$	$\exp\{-\kappa d^2\}$	40	3601	0.5088031	0.0000000	0.2474000
Original	$(1+d^{\kappa})^{-1}$	100	1198	0.0000013	0.009208	0.0000000
Original	$(1+d^{\kappa})^{-1}$	70	1258	0.0000011	0.02533	0.1325000
Original	$(1+d^{\kappa})^{-1}$	40	1687	0.0000569	0.04713	0.6845000
Original	$\exp\{-\kappa d^2\}$	100	1198	0.5026800	0.009208	0.0108500
Original	$\exp\{-\kappa d^2\}$	70	1258	0.4943048	0.004225	0.0294100
Original	$\exp\{-\kappa d^2\}$	40	1687	0.4920137	0.06743	0.1191000
Original (New Seed)	$(1+d^{\kappa})^{-1}$	100	1449	0.0000026	0.0000	0.0000000
Original (New Seed)	$(1+d^{\kappa})^{-1}$	70	1014	0.0000240	0.002046	0.1174000
Original (New Seed)	$(1+d^{\kappa})^{-1}$	40	1609	0.0009413	0.02326	0.6451000
Original (New Seed)	$\exp\{-\kappa d^2\}$	100	1397	0.5100904	0.0007158	0.0005900
Original (New Seed)	$\exp\{-\kappa d^2\}$	70	1393	0.4910087	0.01579	0.05212
Original (New Seed)	$\exp\{-\kappa d^2\}$	40	1620	0.4991936	0.03086	0.1722

Table 4.3: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: data-set, null hypothesis tested and estimated expected p-values from the infection link residuals test, LLR (full likelihood) and LLR (partial likelihood)

increasing the amount of the augmented data which is updated in each sweep.

4.4.2 Results

Estimates of the expected p-values obtained from all three tests over all the data-sets are shown in Table 4.3. These results are plotted in bar-charts in Figures 4.4.1, 4.4.2, 4.4.3, 4.4.4, and 4.4.5.

The ILR test appears to outperform the LLR tests when the fitted model is very different to the actual kernel in most cases although the full LLR still detects a substantial level of mis-specification in a large number of circumstances, and the partial likelihood LLR test detects discrepancy in several of the cases. The ability of the tests to detect mis-specification of the spatial kernel decreased as the epidemics were simulated up to shorter time periods such that a lower proportion of hosts became symptomatically infected. Evidence for this is, when the power law kernel was fitted (and the data was generated from an exponential kernel)

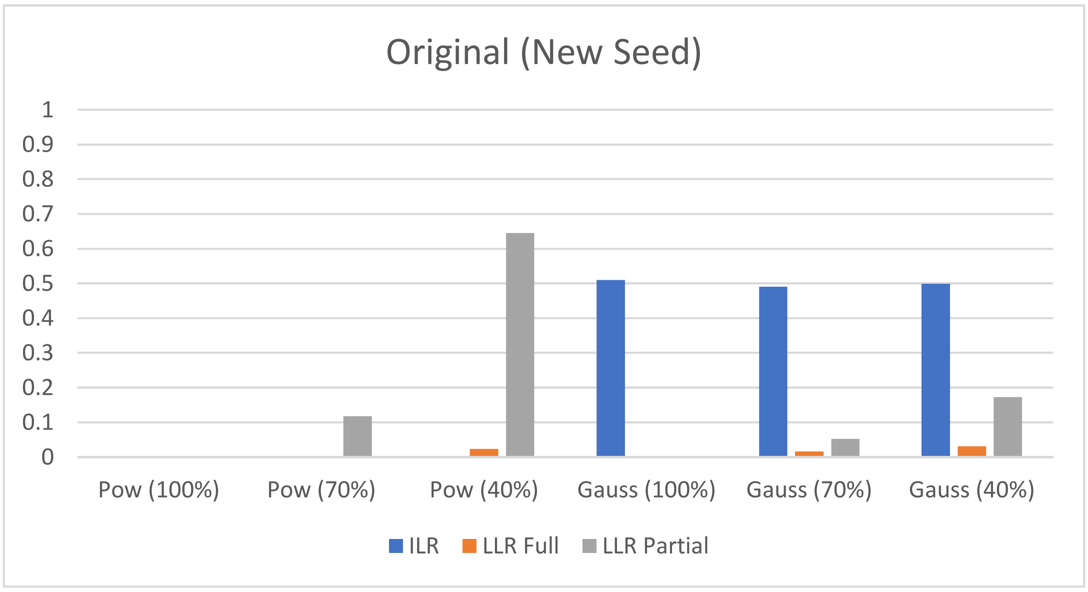

Figure 4.4.1: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: Bar chart of the expected posterior p-values obtain for the data set generated with the original parameters, but with a new random seed for the coordinates of the hosts, where "Pow" denotes a power law kernel was fitted and "Gauss" denotes a Gaussian kernel was fitted. The simulated data was observed up to the time such that a set percentage of the population became infectious. This percentage is in brackets.

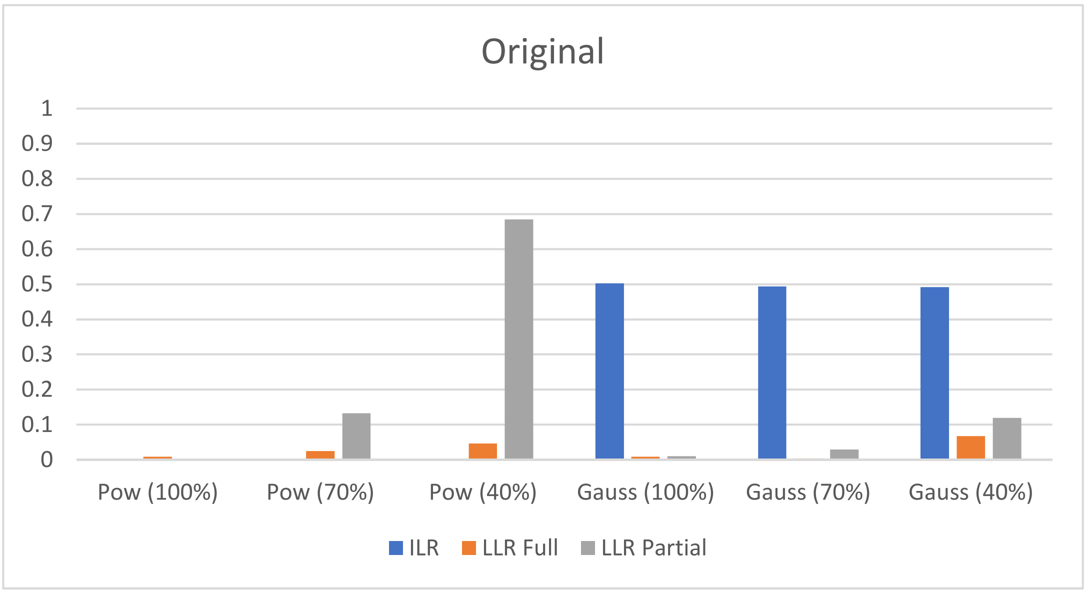

Figure 4.4.2: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: Bar chart of the expected posterior p-values obtain for the data set generated with the original parameters, where "Pow" denotes a power law kernel was fitted and "Gauss" denotes a Gaussian kernel was fitted. The simulated data was observed up to the time such that a set percentage of the population became infectious. This percentage is in brackets.

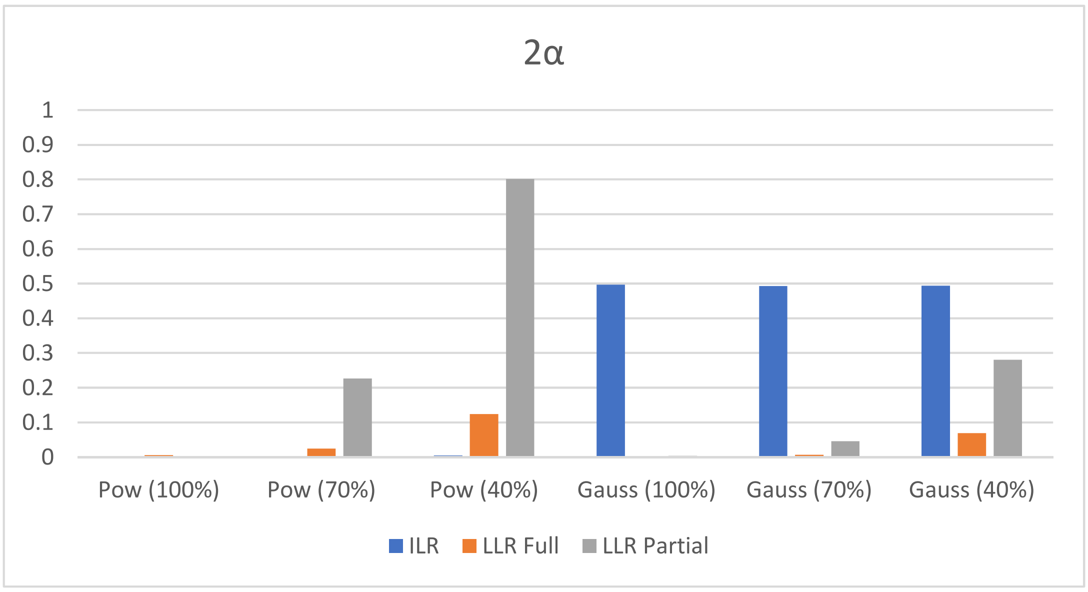

Figure 4.4.3: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: Bar chart of the expected posterior p-values obtain for the data set $\alpha \times 2$, where "Pow" denotes a power law kernel was fitted and "Gauss" denotes a Gaussian kernel was fitted. The simulated data was observed up to the time such that a set percentage of the population became infectious. This percentage is in brackets.

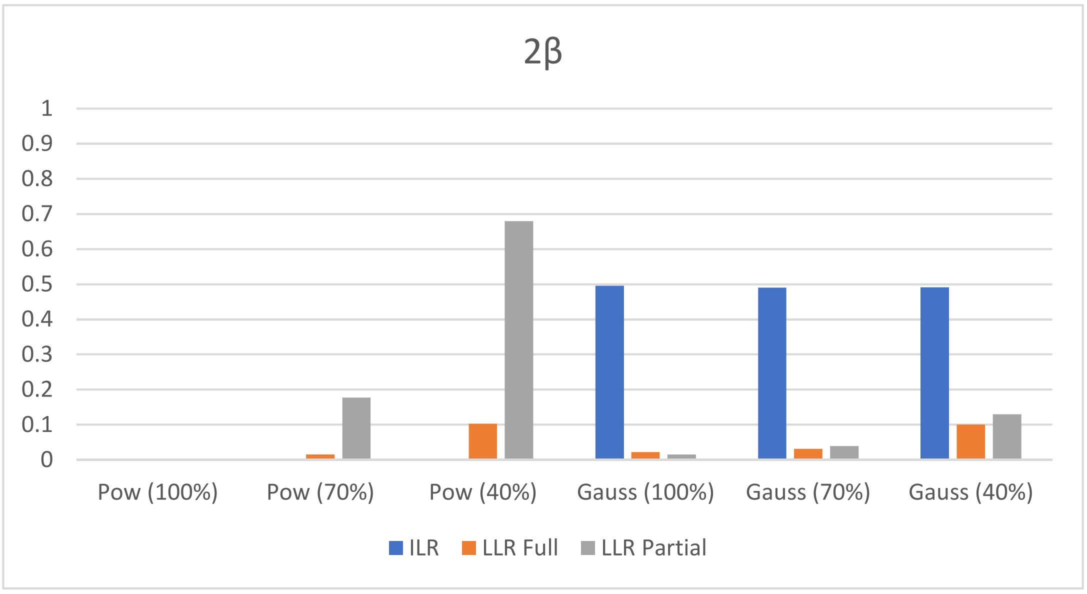

Figure 4.4.4: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: Bar chart of the expected posterior p-values obtain for the data set $\beta \times 2$, where "Pow" denotes a power law kernel was fitted and "Gauss" denotes a Gaussian kernel was fitted. The simulated data was observed up to the time such that a set percentage of the population became infectious. This percentage is in brackets.

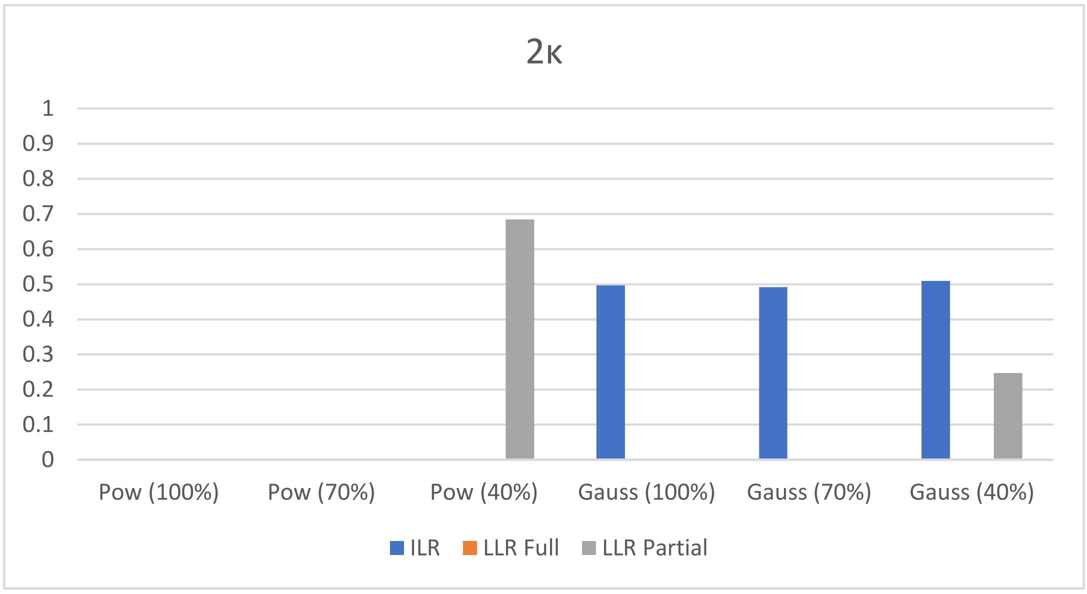

Figure 4.4.5: Comparison of Latent Likelihood Ratio (LLR) test to Infection Link Residuals test: Bar chart of the expected posterior p-values obtain for the data set $\kappa \times 2$, where "Pow" denotes a power law kernel was fitted and "Gauss" denotes a Gaussian kernel was fitted. The simulated data was observed up to the time such that a set percentage of the population became infectious. This percentage is in brackets.

over all datasets, the infection link residuals test detects mis-specification of spatial kernel; the posterior expected p-value produced by the ILR tests all are less than 0.01 indicating that a lot of discrepancy was detected between the fitted model (power law kernel) and the data (generated from an exponential kernel). The latent likelihood ratio tests were also able to detect mis-specification in many cases: in the epidemic datasets which were observed until all the hosts became symptomatic infectious, where a power law spatial kernel was fitted to the data generated from an exponential kernel, indicated by the posterior expected p-values being less than 0.05. These posterior expected p-values (in the $\alpha \times 2$ and $\beta \times 2$ datasets and data generated with the original parameters) are larger than those obtained from the ILR tests, showing that the LLR tests appear to pick up less of the discrepancy than the infection link residuals test. The ability of the LLR tests to detect mis-specification of the spatial kernel is lessened as the epidemic is observed for a shorter period of time. When the time that the epidemic is observed is shortened, the expected posterior p-values increase to the point that they are above 0.1, in the $\alpha \times 2$ and $\beta \times 2$ datasets with the full likelihood LLR test. The partial LLR was able to detect mis-specification when there was high amounts of data but the ability of the partial LLR tests to detect discrepancy falls away much faster than the full likelihood LLR tests with the partial LLR unable to detect substantial discrepancy between the fitted model and the data in the $\alpha \times 2$ and $\beta \times 2$ datasets where the simulated data was generated until 40% of all hosts became symptomatically infected. The expected posterior p-values produced were approximately around 0.7 to 0.8 in the cases of the partial likelihood LLR tests.

In the run results in Table 4.3, the LLR tests outperform the ILR test when the fitted kernel is similar to the actual kernel. The runs with the Gaussian kernel show the relative ability of the LLR and ILR test to detect mis-specification of the spatial kernel when there is less of a difference between the fitted kernel and actual kernel. From Table 4.3, it can be seen that the ILR failed to detect mis-specification of the spatial kernel in all cases where a Gaussian kernel was fitted to data generated from an exponential kernel, with obtained expected posterior p-values being around 0.5. In contrast, the LLR tests produced small p-values, demonstrating that the LLR tests were able to detect substantial misfit between the fitted model and simulated data. The p-values obtained from the LLR tests increased as the truncation time of the data decreased, reflecting the decreasing amount of information available from which to determine the adequacy of the spatial kernel. The p-values obtained from the partial likelihood LLR test increased at a faster rate than the p-values obtained from the full likelihood LLR tests indicating that the full LLR is more able to detect model mis-specification at an earlier time in the epidemic. A set of runs were performed with a new random seed for the XY coordinates of the hosts' spatial locations. The similar results from these runs verify that the patterns hold over many datasets and is not specific to a single random seed.

Also included in this section are tables of parameter estimates obtained from the RJMCMC runs (see Tables 4.4, 4.5, and 4.6). These show how the parameter posterior distributions adapt in order to fit the mis-specified models, and demonstrate the issues faced in epidemic modelling when the transitions to certain states are unobserved. The posterior distribution of the parameters and the augmented data are obtained, and are also used to calculate the test statistics used to determine model fit. Hence, those intending to test model fit of such models should be aware of this reinforcement effect.

4.4.3 Conclusions and Discussion

The results from the exploratory runs and the verification runs suggest that the infection link residual test seems less effective than the LLR tests at detecting model mis-specification when models are similar. The infection link residuals test failed to find significant discrepancy in the runs where the null hypothesis was a Gaussian kernel. This may be because the Gaussian kernel is quite similar to an exponential kernel which is the actual kernel that the data is generated from. The ability of the latent likelihood ratio tests to detect discrepancy when the actual

Data-set	Total % Population Infectious	H_0	α		β		κ	
			Mean	S.D	Mean	S.D	Mean	S.D
$\alpha \times 2$	100	$(1+d^{\kappa})^{-1}$	0.00133	0.0003334	3512	506.4	2.392	0.02861
$\alpha \times 2$	70	$(1+d^{\kappa})^{-1}$	0.0006492	0.0002724	2776	443.8	2.369	0.03153
$\alpha \times 2$	40	$(1+d^{\kappa})^{-1}$	0.0006613	0.0002828	1796	374.7	2.284	0.04026
$\alpha \times 2$	100	$\exp\{-\kappa d^2\}$	0.0029139	0.0004486	0.9023087	0.09043	0.0001535	0.00000898
$\alpha \times 2$	70	$\exp\{-\kappa d^2\}$	0.002556	0.0004495	0.9037788	0.109	0.0001548	0.00001012
$\alpha \times 2$	40	$\exp\{-\kappa d^2\}$	0.0022339	0.0007029	0.8149542	0.5498334	0.0001448	0.0002132
$\beta \times 2$	100	$(1+d^{\kappa})^{-1}$	0.0004442	0.0001923	4794	635.7	2.414	0.02534
$\beta \times 2$	70	$(1+d^{\kappa})^{-1}$	0.0003817	0.0001863	3678	559.2	2.388	0.02941
$\beta \times 2$	40	$(1+d^{\kappa})^{-1}$	0.0004015	0.0001984	2540	457.7	2.312	0.03489
$\beta \times 2$	100	$\exp\{-\kappa d^2\}$	0.0012901	0.0003207	1.0636219	0.141	0.0001178	0.000007319
$\beta \times 2$	70	$\exp\{-\kappa d^2\}$	0.0011284	0.0003027	0.9717919	0.149	0.0001136	0.000007957
$\beta \times 2$	40	$\exp\{-\kappa d^2\}$	0.001087	0.0003053	0.8408861	0.1646	0.0001008	0.000008752
$\kappa \times 2$	100	$(1+d^{\kappa})^{-1}$	0.0008384	0.00004733	2894	426.3	2.673	0.0357
$\kappa \times 2$	70	$(1+d^{\kappa})^{-1}$	0.0007108	0.00006799	2538	401.9	2.644	0.03832
$\kappa \times 2$	40	$(1+d^{\kappa})^{-1}$	0.0005803	0.0001156	2149	384.4	2.604	0.04290
$\kappa \times 2$	100	$\exp\{-\kappa d^2\}$	0.0010206	0.0000484	0.9636156	0.07927	0.0006301	0.00003171
$\kappa \times 2$	70	$\exp\{-\kappa d^2\}$	0.0010663	0.00007045	0.9170497	0.08298	0.0006164	0.00003415
$\kappa \times 2$	40	$\exp\{-\kappa d^2\}$	0.0012108	0.0001243	0.9734283	0.1103	0.0006277	0.00004283
Original	100	$(1+d^{\kappa})^{-1}$	0.0003585	0.000178	2558	476.7	2.378	0.03547
Original	70	$(1+d^{\kappa})^{-1}$	0.0003146	0.0001575	3533	525.7	2.434	0.02942
Original	40	$(1+d^{\kappa})^{-1}$	0.000696	0.000193	4313	561.4	2.455	0.02589
Original	100	$\exp\{-\kappa d^2\}$	0.0013476	0.0002592	0.8482083	0.08723	0.0001498	0.000008934
Original	70	$\exp\{-\kappa d^2\}$	0.0010269	0.0002521	0.9342897	0.1178	0.0001527	0.00001006
Original	40	$\exp\{-\kappa d^2\}$	0.0009899	0.000265	0.9450705	0.1553	0.0001521	0.00001294
Original (New Seed)	100	$(1+d^{\kappa})^{-1}$	0.001213	0.0002561	3731	540.5	2.41	0.02903
Original (New Seed)	70	$(1+d^{\kappa})^{-1}$	0.0009564	0.0003077	3114	523.6	2.377	0.04164
Original (New Seed)	40	$(1+d^{\kappa})^{-1}$	0.001012	0.0003282	2129	431.8	2.296	0.04251
Original (New Seed)	100	$\exp\{-\kappa d^2\}$	0.0016983	0.0002973	0.6972467	0.06671	0.0001295	0.00000753
Original (New Seed)	70	$\exp\{-\kappa d^2\}$	0.0016752	0.0003794	0.7186071	0.08508	0.0001275	0.000008302
Original (New Seed)	40	$\exp\{-\kappa d^2\}$	0.0018358	0.0004172	0.8079134	0.122	0.0001332	0.00001107

Table 4.4: Table of Posterior Means and Standard Deviation for the verification runs for α, β, and κ

and mis-specified kernel are very similar could be attributed to the fact that the latent likelihood ratio tests require a specific alternative hypothesis, whilst the infection link residuals test does not require a specific alternative hypothesis.

From the results in Table 4.3, observe that in all the datasets in which a Gaussian kernel was fitted, the full likelihood LLR test was able to detect substantial discrepancy between the fitted Gaussian kernel and the actual exponential kernel. The partial likelihood LLR test was able to detect substantial discrepancy in data-sets where the end observation time was set such that more than 70% and 40% of the hosts became infectious. The ability of the partial likelihood LLR test fell away more quickly than that of the full likelihood LLR test especially in the $2\times\alpha$ and $2\times\kappa$ runs. As to why this is the case, a possible explanation is as follows:

- If we only consider the order of infection times, and not the infection event times themselves, an increase in α is the same as a decrease in β. So the effect of secondary infection in the $2 \times \alpha$ runs is less than that in the data using the original parameters.

Data-set	Total % Population Infectious	H_0	μ_E		σ_E^2	
			Mean	S.D	Mean	S.D
$\alpha \times 2$	100	$(1+d^{\kappa})^{-1}$	4.805	0.1031	1.956	0.2294
$\alpha \times 2$	70	$(1+d^{\kappa})^{-1}$	4.523	0.1189	1.741	0.2386
$\alpha \times 2$	40	$(1+d^{\kappa})^{-1}$	4.451	0.1521	1.805	0.344
$\alpha \times 2$	100	$\exp\{-\kappa d^2\}$	5.042126	0.1102	2.4812384	0.2898
$\alpha \times 2$	70	$\exp\{-\kappa d^2\}$	4.9593773	0.1377	2.3839773	0.3476
$\alpha \times 2$	40	$\exp\{-\kappa d^2\}$	4.843329	0.1968621	2.1531098	0.4522831
$\beta \times 2$	100	$(1+d^{\kappa})^{-1}$	4.289	0.1153	1.925	0.2174
$\beta \times 2$	70	$(1+d^{\kappa})^{-1}$	4.041	0.1357	1.819	0.2626
$\beta \times 2$	40	$(1+d^{\kappa})^{-1}$	4.022	0.1633	1.912	0.352
$\beta \times 2$	100	$\exp\{-\kappa d^2\}$	4.7652099	0.124	2.5839779	0.2968
$\beta \times 2$	70	$\exp\{-\kappa d^2\}$	4.6984782	0.1549	2.4853798	0.3787
$\beta \times 2$	40	$\exp\{-\kappa d^2\}$	4.6826162	0.2359	2.6409302	0.6064
$\kappa \times 2$	100	$(1+d^{\kappa})^{-1}$	4.945	0.1001	2.816	0.2872
$\kappa \times 2$	70	$(1+d^{\kappa})^{-1}$	4.929	0.1121	2.872	0.339
$\kappa \times 2$	40	$(1+d^{\kappa})^{-1}$	4.969	0.1349	2.702	0.3944
$\kappa \times 2$	100	$\exp\{-\kappa d^2\}$	4.9826195	0.09697	2.64989	0.2738
$\kappa \times 2$	70	$\exp\{-\kappa d^2\}$	4.979419	0.1082	2.6699676	0.3042
$\kappa \times 2$	40	$\exp\{-\kappa d^2\}$	5.0850028	0.1401	2.8226393	0.4221
Original	100	$(1+d^{\kappa})^{-1}$	4.42	0.162	2.057	0.3825
Original	70	$(1+d^{\kappa})^{-1}$	4.473	0.1243	2.06	0.2758
Original	40	$(1+d^{\kappa})^{-1}$	4.579	0.1075	2.236	0.2442
Original	100	$\exp\{-\kappa d^2\}$	4.793396	0.1173	2.7646615	0.3458
Original	70	$\exp\{-\kappa d^2\}$	4.8615001	0.1504	2.9812697	0.4669
Original	40	$\exp\{-\kappa d^2\}$	4.9118098	0.1972	2.8702635	0.5652
Original (New Seed)	100	$(1+d^{\kappa})^{-1}$	4.862	0.1082	2.362	0.2636
Original (New Seed)	70	$(1+d^{\kappa})^{-1}$	4.756	0.1305	2.26	0.2975
Original (New Seed)	40	$(1+d^{\kappa})^{-1}$	4.66	0.1573	1.934	0.3628
Original (New Seed)	100	$\exp\{-\kappa d^2\}$	4.9423912	0.1094	2.572538	0.3046
Original (New Seed)	70	$\exp\{-\kappa d^2\}$	4.9488673	0.1389	2.4965248	0.3723
Original (New Seed)	40	$\exp\{-\kappa d^2\}$	4.9545913	0.1762	2.5261648	0.4714

Table 4.5: Table of Posterior Means and Standard Deviation for the verification runs for μ_E and σ_E^2

Data-set	Total % Population Infectious	H_0	μ_I		σ_I^2	
			Mean	S.D	Mean	S.D
$\alpha \times 2$	100	$(1+d^{\kappa})^{-1}$	1.801	0.03027	0.9134	0.04996
$\alpha \times 2$	70	$(1+d^{\kappa})^{-1}$	1.816	0.03786	0.9332	0.06434
$\alpha \times 2$	40	$(1+d^{\kappa})^{-1}$	1.825	0.05211	0.8917	0.08827
$\alpha \times 2$	100	$\exp\{-\kappa d^2\}$	1.8004701	0.03029	0.9131307	0.05003
$\alpha \times 2$	70	$\exp\{-\kappa d^2\}$	1.8152959	0.03783	0.9321667	0.06426
$\alpha \times 2$	40	$\exp\{-\kappa d^2\}$	1.8247609	0.0520868	0.8913357	0.0879867
$\beta \times 2$	100	$(1+d^{\kappa})^{-1}$	1.8	0.03024	0.9131	0.04983
$\beta \times 2$	70	$(1+d^{\kappa})^{-1}$	1.796	0.03717	0.883	0.06131
$\beta \times 2$	40	$(1+d^{\kappa})^{-1}$	1.825	0.05158	0.8927	0.0875
$\beta \times 2$	100	$\exp\{-\kappa d^2\}$	1.8002562	0.03023	0.9128844	0.04977
$\beta \times 2$	70	$\exp\{-\kappa d^2\}$	1.7959016	0.03717	0.8831906	0.06121
$\beta \times 2$	40	$\exp\{-\kappa d^2\}$	1.8250691	0.05174	0.8928353	0.0876
$\kappa \times 2$	100	$(1+d^{\kappa})^{-1}$	1.801	0.0302	0.9132	0.04992
$\kappa \times 2$	70	$(1+d^{\kappa})^{-1}$	1.835	0.03691	0.9523	0.06244
$\kappa \times 2$	40	$(1+d^{\kappa})^{-1}$	1.851	0.04849	0.9248	0.08094
$\kappa \times 2$	100	$\exp\{-\kappa d^2\}$	1.8003755	0.03025	0.9130562	0.04992
$\kappa \times 2$	70	$\exp\{-\kappa d^2\}$	1.8351697	0.03694	0.9522412	0.0625
$\kappa \times 2$	40	$\exp\{-\kappa d^2\}$	1.8512757	0.04859	0.9250943	0.08113
Original	100	$(1+d^{\kappa})^{-1}$	1.798	0.05017	0.8894	0.08347
Original	70	$(1+d^{\kappa})^{-1}$	1.821	0.03738	0.9156	0.06274
Original	40	$(1+d^{\kappa})^{-1}$	1.8	0.0303	0.913	0.04998
Original	100	$\exp\{-\kappa d^2\}$	1.8003004	0.03019	0.9129414	0.04981
Original	70	$\exp\{-\kappa d^2\}$	1.8210957	0.03744	0.9158055	0.06292
Original	40	$\exp\{-\kappa d^2\}$	1.797982	0.05013	0.8895275	0.08353
Original (New Seed)	100	$(1+d^{\kappa})^{-1}$	1.752	0.02889	0.8338	0.04531
Original (New Seed)	70	$(1+d^{\kappa})^{-1}$	1.771	0.03564	0.8385	0.05709
Original (New Seed)	40	$(1+d^{\kappa})^{-1}$	1.793	0.04971	0.7834	0.07785
Original (New Seed)	100	$\exp\{-\kappa d^2\}$	1.7518828	0.0289	0.8337618	0.04526
Original (New Seed)	70	$\exp\{-\kappa d^2\}$	1.7715113	0.03561	0.8385418	0.05685
Original (New Seed)	40	$\exp\{-\kappa d^2\}$	1.7935544	0.04976	0.7836434	0.0782

Table 4.6: Table of Posterior Means and Standard Deviation for the verification runs for μ_I and σ_I^2

- Again, if we only consider the order of the infection times and not the infection event times themselves, an increase in κ will lower the effect of secondary infection, as the spatial kernel is not normalised in any way.
- Thus, the cases where the partial likelihood LLR test's ability to detect discrepancy falls away fast seems to be the cases where there is less effect of secondary infection.
- This lowered effect of secondary infection in the data that was generated could be the cause of the larger decrease in the ability to detect discrepancy from the fitted model and the data as the epidemic was observed for a shorter interval of time:
  - At the start of each simulated epidemic, as there are relatively few infectious hosts, the majority of infections will be primary infections.
  - At the peak of each simulated epidemic, as there are many infectious hosts, there will be a relatively large amount of secondary infections compared to primary infections.
  - If the order of infection times alone is used to determine model adequacy, from a sequence of primary infections, it is impossible to determine whether the model is adequate but not.
  - If the infection times are used to determine model adequacy (in addition to the order of the infection times, as in the full likelihood LLR test), it is possible to check the adequacy of the model from a stream of primary infections, since the rate of primary infections should be plausible if the model is adequate.
  - Since at the start of each simulated epidemic, the majority of infections are primary infections, the full likelihood LLR test would be more able to check model adequacy than the partial likelihood LLR.
  - This effect is amplified by the lowering of the effect of the secondary infection in the simulated data.

- An example in which this argument can be intuitively understood is as follows: consider a simulated epidemic in which there is only primary infection and no secondary infection. The partial likelihood LLR would be unable to detect whether the model is adequate or not as it only uses information on the order of the infections. Full likelihood LLR would be able to detect whether the model is adequate or not as it uses the event times as well as the order of the event times.

An implication of this is that the full likelihood LLR test still remains a more robust method for detecting model inadequacy than the partial likelihood LLR test, at least in the case of comparing between similar kernels.

The infection link residuals test seems more effective that the LLR tests at detecting model mis-specification when the models are very different. In all the cases where a power law kernel was fitted to data generated from an exponential kernel, the ILR was able to detect substantial discrepancy between the data and the model. The full likelihood LLR also detected significant discrepancy between model and data in all cases apart from one, which still had a small expected posterior p-value. These expected posterior p-values were not as small as those obtained for the ILR test, and a possible explanation for this is that the ILR test has a vague alternative hypothesis whilst the LLR tests have a specific alternative hypothesis, so are looking for discrepancy between model and data which conforms to this specific alternative hypothesis. It was also observed that the partial likelihood LLR tests produced similar posterior expected p-values when the simulated data was generated until a time such that all the population became infectious. As this time truncation was reduced, the ability of the partial likelihood LLR test diminished to detect discrepancy much faster than that of the full likelihood LLR test, possibly for the same reasons given as in the runs in which the Gaussian spatial kernel was fitted to data from an exponential kernel. It appears that in some cases, for example in the exploratory runs, the partial likelihood LLR test was able to detect discrepancy between models and data in the run in which a Cauchy kernel was fitted to data generated from an exponential kernel, and the data was simulated until the time that 40% of the population became infectious. This indicates there may be a point at which the difference between the fitted kernel and the actual kernel becomes so large that the partial likelihood LLR test is able to detect mis-specification of spatial kernel, even at early stages in epidemic. However, it seems from the runs where a power law kernel was fitted to data generated from an exponential kernel that the full likelihood LLR test remains the most robust method for detecting mis-specification of spatial kernel.

It appears that because the full likelihood LLR takes into account infection times as well as order whilst the partial LLR only considers infection order, the full likelihood ratio test was able to detect kernel mis-specification in cases where the partial likelihood LLR was not. From the above proposed explanation of why the ability to detect discrepancy in spatial kernel falls away faster for the partial likelihood LLR compared to the full likelihood LLR, it appears that taking the infection times into account when comparing between two similar kernels is important for the detection of mis-specification of spatial kernel. From the posterior parameter means and variances of α given in Table 4.4, it can be seen that the primary infection rate often compensates for a kernel in which the tail is too short or too long. Since the majority of infections during the early phases of these simulated epidemics were primary infections, the full likelihood LLR test could detect mis-specification unlike the partial likelihood LLR test which only took into account the order of infections. It appears that the ILR test was able to detect mis-specification, since its alternative hypothesis is non-specific, and therefore takes into account all information on all discrepancy, whilst the LLR test's specific alternative hypothesis reduced the information available to the test, and infection time information was needed in addition to the infection order information.

It is very difficult to analytically derive a result of the relative power of the two methods, as the likelihoods and models are complex. In addition, comparison with the infection link residual test is difficult, as the infection link residuals test relies on the Anderson-Darling test to detect non-uniformity of the infection link residuals. This means that the power of the infection link residuals test is limited by the test used to detect non-uniformity of the residuals. At the current date of writing there appears to be no literature which has derived results for the power of the Anderson-Darling test which are relevant to this problem.

An area of investigation in the future is the derivation of the distribution of the test statistic for the latent likelihood ratio test. In the case of nested models, it is known asymptotically to be a chi-square distribution. But in the case of non-nested models, Monte Carlo simulation has been used in this case to obtain an estimate of the posterior expected p-value, which is very computationally intensive, because this algorithm cannot be parallelised. In fact most of the computation time is probably due to the regeneration of an epidemic with the parameters from the alternative model. This computational intensive makes it unfeasible to calculate an estimate of the p-value each time a test statistic is calculated. Instead, the dataset is regenerated under the fitted model, and a test that is recalculated and it is determined whether this value is larger than the obtained test statistic. As a result, a series of zeros and ones are obtained, which allow the calculation of the expected posterior p-value. However, it would be much more informative to calculate the exact p-values of each calculator test statistic, which would then allow the posterior distribution of the p-value to be obtained, which would be much more informative. It is known that in many of the cases obtained here that the posterior expect p-value is very low, but it would be informative to know the shape of the distribution and exactly how much of the distribution is below 0.05. Further insight can be gained into the relative performance of each test from how the shape of the posterior distribution of the p-value changes with different datasets.

Regarding the computational intensiveness of the expected posterior p-value calculation, the majority of these computational difficulties have been surmounted by the implementation of likelihood function on the graphics processor, which will be described in the following chapter. Without these innovations, the runs performed in this thesis would have been extremely time-consuming, or impossible.

However, these innovations allow these runs to be performed in under a day. Since the likelihood function in central to many of the calculations for the iterative algorithms, including both the Monte Carlo methods and the maximum-likelihood estimation, and acceleration in this produces quite a gaining speed makes these methods feasible in a normal amount of time.

Regarding further investigation, a possible route for further investigation is whether the infection link residuals test can be strengthened to compete against the LLR tests, without using a specific alternative hypothesis like the LLR tests. As mentioned above, one of the reasons why the full likelihood LLR test was more able to detect discrepancy then the partial likelihood LLR test in many circumstances was that it incorporated information about the transition times as well as transition order. In the original paper [111], there were four different proposed residuals. The first of those two residuals $\tilde{\boldsymbol{r}}_1$ (known as the *exposure time residuals*) regarding the S to E transitions account for the infection times but not infection order, and is effectively a population level Sellke threshold. The second of those were the infection link residuals $\tilde{\boldsymbol{r}}_2$. Perhaps by combining these two tests in some way, a more powerful test could be obtained, although the problem of reinforcement could occur.

Another approach could be to focus whatever discrepancy that there has been obtained, discarding even more information to reduce the effect of the reinforcement: the infection link residual could be modified to create a new type of residual which is the absolute value of 0.5 minus infection link residual. Under the null hypothesis this is obviously uniformly distributed. Further investigation could be done to investigate whether this is more capable of detecting discrepancy between the null hypothesis than the existing infection link residuals test.

A key benefit of using the functional-models and generalised residuals representation of the epidemic model is that it can be used to produce tests which are focused on different aspects of mis-specification. As long as the sampling distribution of such a representation is exactly that of the model, any representation of the epidemic model can be chosen. The existing infection link residuals method is essentially reverse engineering the Gillespie-type algorithm, splitting the step where the next infection is chosen into two steps: choosing the infective host and infected host. Alternative representations of choosing how the infection link is chosen can be formulated to detect anisotropy, for example. This will be investigated in Chapter 6, where a test will be formulated to detect anisotropy.

In many situations, it may be important to determine whether there is anisotropy present in how the infection spreads. The detection of such anisotropy could indicate that there is an effect of wind or some other directional effects which has not been taken account of by the model that has been fitted.

4.5 A Discussion of Reinforcement

Here we discuss how when using LLRT tests it is conceivable that the greater the imputed information on which the test is based, the poorer the performance of the test. This argument was originally presented in [68] by Gibson and Streftaris.

Definition 18. Let M_0 and M_1 be two models under comparison with a latent likelihood ratio test. Let π_0 and π_1 be the sampling densities under M_0 and M_1 respectively. In the latent likelihood ratio test, x is imputed using posterior distribution under M_0, $\pi_0(x|y)$, and from this imputed x, the likelihood ratio test is used on the imputed data to obtain p-value $p(x)$, yielding distribution $\pi_0(p(x)|y)$. Let the *power* for a latent test be defined as:

$$\beta_x = E(\pi_0(p(x) < \alpha|y)|M_1)$$

Remark 19. If M_0 includes a prior with all belief at $\theta = \theta_0$, and:

$$M_0 : \theta = \theta_0$$
$$M_1 : \theta = \theta_1$$

When $x \equiv y$, there is no unobserved data, and the latent likelihood ratio test reduces to the classical likelihood ratio test, and $\beta_x = \beta_y$ which is the classical power of the likelihood ratio test, which is the UMP test of M_0 vs M_1, by the Neyman-Pearson Lemma.

Proposition 20. *Suppose M_0 includes a prior with all belief at $\theta = \theta_0$, and:*

$$M_0 : \theta = \theta_0$$
$$M_1 : \theta = \theta_1$$

Then $\beta_x \leq \beta_y$.

Proof. In this case, the latent likelihood ratio test can be thought of as a Bayesian's belief of what a frequentist's p-value would be. Furthermore the whole process of performing a latent likelihood ratio test can be seen as a Bayesian imputing the unobserved data to obtain x, then a frequentist performing a classical ratio test with that data at level α. The UMP test of level α of M_0 vs. M_1 with data x is the likelihood ratio test. Let $\pi_0^i(x)$ and $\pi_1^i(x)$ be the sampling densities under M_0 and M_1 of imputed x respectively. The test statistic of this test is:

$$\frac{\pi_0^i(x)}{\pi_1^i(x)} \tag{4.5.1}$$

Since $\pi_0^i(x) = \pi_0(x|y)\pi_0(y)$, and $\pi_0^i(x) = \pi_0(x|y)\pi_1(y)$, (4.5.1) can be expressed as:

$$\begin{aligned} &\frac{\pi_0(x|y)\pi_0(y)}{\pi_0(x|y)\pi_1(y)} \\ =&\frac{\pi_0(y)}{\pi_1(y)} \end{aligned}$$

This is the same test statistic as a likelihood ratio test applied to y, thus, the power of the UMP test statistic is β_y.

However, in a latent likelihood ratio test, the test statistic:

$$\frac{\pi_0(x)}{\pi_1(x)} = \frac{\pi_0(x|y)\pi_0(y)}{\pi_1(x|y)\pi_1(y)}$$

is used instead. This is not the same as:

$$\frac{\pi_0^i(x)}{\pi_1(x)}$$

This is not the test statistic of (4.5.1), and thus by the Neyman-Pearson lemma, has less power as a test statistic. Thus, for any specified α,

$$P(p(x) < \alpha|M_1) \leq P(p(y) < \alpha|M_1) = \beta_y$$

Since:

$$\begin{aligned} P(p(x) < \alpha|M_1) &= \int \pi_0(p(x) < \alpha|y)\pi_1(y)dy \\ &= E(\pi_0(p(x) < \alpha|y)|M_1) \\ &= \beta_x \end{aligned}$$

Thus:

$$\beta_x = E(\pi_0(p(x) < \alpha|y)|M_1) \leq E(\pi_0(p(x) < \alpha|y)|M_1) = \beta_y$$

□

This shows that the power of a latent likelihood ratio test applied to x cannot exceed that of a likelihood ratio test applied to observed data y. Furthermore:

suppose M_0 includes a prior $\pi(\theta)$, and:

$$\begin{aligned} M_0 :& \theta = \theta_0 \\ M_1 :& \theta = \theta_1 \end{aligned}$$

Then $\beta_{x,\theta} \leq \beta_{y,\theta}$.

The above result applies to only a very simple case of latent likelihood ratio testing, but demonstrates that the loss of power, or ability to discern whether there is substantial discrepancy from the actual model, is due to the use of $\pi_0(x|y)$, the imputation of the unobserved data under M_0. As a result one can argue that as the amount of imputed data increases, the latent likelihood ratio test loses more ability to discern whether there is substantial discrepancy from the actual model:

Consider a situation where simple hypotheses M_0 and M_1 are being compared.

Suppose that $y = f(x)$ and $x = g(z)$ such that z has more information than x, z includes information not observed. Let π_0 be the sampling density of quantities under M_0 and $\pi_1$be the sampling density of quantities under M_1.

Suppose we impute z. Then the optimal test statistic is

$$\frac{\pi_0(z)}{\pi_1(z)} = \frac{\pi_0(z)}{\pi_0(z|y)\pi_1(y)} \tag{4.5.2}$$

.

Let π_0^i and π_1^i be the imputed sampling densities under M_0 and M_1 respectively. Then the Kullback-Leibler Divergence between the imputed sampling density of z under M_1,$\pi_1^i(z)$ and the sampling density of z under M_1, denoted $KL(\pi_1^i, \pi_1)$ is determined by the following:

$$
\begin{aligned}
KL(\pi_1^i, \pi_1) &= \int \pi_1^i(z) \log\left(\frac{\pi_1^i(z)}{\pi_1(z)}\right) dz \qquad (4.5.3)\\
&= \int \pi_1^i(z) \log\left(\frac{\pi_0(z|x)\pi_0(x|y)\pi_1(y)}{\pi_1(z|x)\pi_1(x|y)\pi_1(y)}\right) dz\\
&= \int \pi_1^i(z) \log\left(\frac{\pi_0(z|x)\pi_0(x|y)}{\pi_1(z|x)\pi_1(x|y)}\right) dz\\
&= \int \pi_1^i(z) \left(\log\left(\frac{\pi_0(x|y)}{\pi_1(x|y)}\right) + \log\left(\frac{\pi_0(z|x)}{\pi_1(z|x)}\right)\right) dz\\
&= \int \pi_1^i(z) \left(\log\left(\frac{\pi_0(x|y)}{\pi_1(x|y)}\right)\right) dz\\
&+ \int \pi_1^i(z) \left(\log\left(\frac{\pi_0(z|x)}{\pi_1(z|x)}\right)\right) dz\\
&= \int \pi_0(z|x)\pi_0(x|y)\pi_1(y) \left(\log\left(\frac{\pi_0(x|y)}{\pi_1(x|y)}\right)\right) dz\\
&+ \int \pi_0(z|x)\pi_0(x|y)\pi_1(y) \left(\log\left(\frac{\pi_0(z|x)}{\pi_1(z|x)}\right)\right) dz\\
&= \int \pi_0(z|x)\pi_0(x|y)\pi_1(y) \left(\log\left(\frac{\pi_1(y)\pi_0(x|y)\pi_0(z|x)}{\pi_1(y)\pi_1(x|y)\pi_0(z|x)}\right)\right) dz \qquad (4.5.4)\\
&+ \int \pi_0(z|x)\pi_0(x|y)\pi_1(y) \left(\log\left(\frac{\pi_1(y)\pi_0(x|y)\pi_0(z|x)}{\pi_1(y)\pi_0(x|y)\pi_1(z|x)}\right)\right) dz \qquad (4.5.5)
\end{aligned}
$$

The integral (4.5.4) is equal to $KL(\pi_1^i(z), \pi_1(y)\pi_1(x|y)\pi_0(z|x))$.

The integral (4.5.5) is another K-L divergence and hence greater than 0. Hence, from (4.5.3),

$$KL(\pi_1^i, \pi_1) \geq KL(\pi_1^i(z), \pi_1(y)\pi_1(x|y)\pi_0(z|x))$$

There is a larger difference between distributions π_1^i and π_1, than $\pi_1^i(z)$ and $\pi_1(y)\pi_1(x|y)\pi_0(z|x)$.

Recall from (4.5.2) that the test statistic for z is:

$$\frac{\pi_0(z)}{\pi_0(z|y)\pi_1(y)}$$

Suppose that $\pi_1(y)\pi_1(x|y)\pi_0(z|x)$ is used on the denominator of a test statistic.

$$\begin{aligned}
&\frac{\pi_0(z)}{\pi_0(z|y)\pi_1(x|y)\pi_1(y)}\\
&=\frac{\pi_0(z|y)\pi_0(x|y)\pi_0(y)}{\pi_0(z|y)\pi_1(x|y)\pi_1(y)}\\
&=\frac{\pi_0(x)}{\pi_1(x)}
\end{aligned}$$

So the test statistic for z, if z is imputed, has the power as a test applied to x (if x were observed, not imputed), since their test statistics are equivalent.

If both x and z are imputed, when a LLR test is performed, the test statistic used is:

$$\begin{aligned}
&\frac{\pi_0(z)}{\pi_1(z)}\\
=&\frac{\pi_0(z|y)\pi_0(x|y)\pi_0(y)}{\pi_0(z|y)\pi_0(x|y)\pi_1(y)}
\end{aligned}$$

The test statistic $\frac{\pi_0(z|y)\pi_0(x|y)\pi_0(y)}{\pi_0(z|y)\pi_1(x|y)\pi_1(y)}$, has a denominator which is “closer” to $\pi_1^i(z)$ than $\pi_1(z)$, and has the same power as a test applied to an unimputed x. Thus, $\frac{\pi_0(z|y)\pi_0(x|y)\pi_0(y)}{\pi_0(z|y)\pi_1(x|y)\pi_1(y)}$, the test statistic with less imputation (only z imputed) than the test statistic $\frac{\pi_0(z)}{\pi_1(z)}$ (where both x and z are imputed), is likely to be more able to detect discrepancy between the actual and fitted model.

Chapter 5

Implementation of Algorithms using Massively Parallel Programming Techniques

In the previous chapters, we have laid out the theoretical framework for model comparison in epidemic models. These methods are computationally intensive, to the extent that the initial test runs on a single thread on a single *Central Processing Unit* (CPU) took an impractical amount of time, rendering the methods initially too computationally difficult to use in a real-world application. This led to the exploration of parallel computation methods to accelerate the computation of the methods detailed in this thesis. Initially, straightforward parallel computation was attempted on the CPU, but this did not yield an adequate speed up. Next, cluster computing or distributed computing was investigated as a way of accelerating the computations, but after some thought it was reasoned that the costs in terms of communications between the CPUs would probably not yield any net acceleration. This led us to consider computation on an accelerator co-processor.

The use of accelerator coprocessors has become popular recently as a power efficient and cost-efficient method [174, 175] of gaining heterogeneous massive parallelism and allows high speed computation on consumer grade hardware.

Massive parallelism (a term coined in the late 1970s and early 80s, for example, [12, 54]) refers to the division of a computational task into many, usually thousands or millions, of parts and processing each part on a processor, the computer being constructed of many processors running in parallel to complete the computational task.

Massive parallelism has usually been considered as a field of programming only utilised on supercomputers or cluster computers. The processors communicate within the supercomputer through special hardware (modern supercomputers consist of several thousand processors), and the processors in the cluster computer communicate through a network. Supercomputers have generally been costly (in monetary terms) to use [8], and cluster computers incur a penalty through the communication of processors through a network [118] (the speed of light is 30cm per nanosecond), making cluster computers suited only to very specific parallel tasks.

The use of accelerator coprocessors has become popular as of recently with *Graphic Processor Unit* (GPU) programming [136] and Intel launching its own coprocessor [88]. The coprocessor unit is used as well as the computer's CPU to run the computational task, which is divided between the CPU and coprocessors. Because the parallel task is divided between two types of processor which are different to each other, this is known as *heterogeneous massive parallelism*. Since all the processors are situated in the same computer, there is less of a penalty when communication is required between the processors.

The GPU is used as a coprocessor in the computations in this thesis, as GPUs are found in all computers. GPU programming used to be a niche field, with programmers having to write their programs in terms of graphical objects. Recently graphics card manufacturers discovered that programmers outside the traditional field of computer graphics were using their processors for numeric computation (for example, [84, 190]), and created their own programming languages to allow programmers to program a graphics card without the need to re-express their algorithms in terms of graphics. One of the dominant programming languages is CUDA C++ [135] which is an extended version of C++ which allows programs to be written for GPUs manufactured by the NVIDIA Corporation. This language has the benefit of being able to interface with existing C++ code without complicated programming. Note however, that GPU programming is different from CPU programming. The constraints on what can be programmed on the GPU are laid out below.

As well as pertaining to our computational requirements, massively parallel heterogeneous computation is relevant and beneficial to the field of epidemic modelling and, arguably, to Bayesian statistics as a whole. In the past, statistics and epidemic modelling have both benefited from increases in computational power. This has mostly been in the form of increases in processor clock speed, making any given computer code run faster without any change in programming. However, the power consumption requirements of increasing the clock speed any further have made any further gain in clock speed impractical [160]. Any increase in computer power has been due to the number of parallel processes running on the same computer. This implies that in order to benefit from any advances in computing power, the same computer code cannot be used [177, 5]. Instead, modifications must be made to take advantage of the parallel processing resources available in the computer. There has been an increasing demand for more computer power as methods and data-sets become larger and more complex. The construction of efficient parallel algorithms for Bayesian computation and model selection, especially in epidemic modelling, becomes of vital importance, allowing researchers to explore more theoretically complex methods and extract information and conclusions from larger data.

5.1 Glossary of technical terms

At this point we find it helpful to provide a glossary of the specialist terms regarding the programming of the GPU as an accelerator coprocessor.

API	Application Programming Interface. Defined in [144] as: "A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service".
architecture	The components of a computer or a computer part and how they are related to each other.
asynchronous	Refers to parallel operations that do not start and/or finish at the same time (and therefore happen independent of each other).
atomic	An operation that cannot be interrupted or interfered with once started and appears to other threads as one indivisible operation.
bandwidth	The maximum rate at which data can be transfered from one component to another.
branch divergence	Where some threads (see *thread*) in a warp (see *warp*) satisfy an "if" statement (branch) and some do not. Considered a drain on performance. Some architectures have the capability built into the hardware to minimise this performance drain.
cache	Defined in [144] as: "An auxiliary memory from which high-speed retrieval is possible.".
clock cycle	The amount of time between successive pulses of the CPU or GPU clock. This is the smallest unit of time in which one processor activity can be performed. [146]

clock speed	Number of clock cycles per second. Usually given in Hertz, for example, 1 GHz is 10^9 clock cycles per second.
coalesced access	Refering to a read or write where consecutive threads read consecutive memory addresses. One of the methods by which memory bandwidth can be used most efficently.
compile	To translate from a high-level language (in this example CUDA C++) to machine code (binary). For example, a compile-time error is a mistake in the code which causes the compiler to be unable to compile the code.
compute capability	"The compute capability of a GPU determines its general specifications and available features." [139]. This is expressed as a number, or a name refering to GPUs of the same architecture e.g. Fermi, Kepler, Maxwell, Pascal, Volta. Synonymous with "architecture".
debug	To modify code to eliminate errors.
device	In GPGPU programming, refers to the GPU.
DRAM	Dynamic random access memory. Also known as the RAM.
GPGPU	General Purpose computing on the GPU. Use of the GPU in non-traditional fields, such as scientific computation.
host	In GPGPU programming, refers to the CPU.
runtime	The period of time when the program is run. For example, a "runtime error" is an error that occurs when the program is running but does not hinder the program from being compiled.
texture	A term originally used in GPU programming to refer to images which are laid (after some transformation) over a

	polygon mesh when creating 3D graphics. In GPGPU pr-gramming this term is often used to refer to texture memory, which is convenient to use in the situation where consecut-ive threads access consecutive memory addresses. Texture memory is also cached on-chip, creating a possible perform-ance benefit.
thread	“A single sequential flow of control within a program.” [30]. Like many computing terms, this term is loosely defined, with many different definitions of "thread" evolving since the 1950s which are similar to the definition given here, but may not be equivalent. Common to all definitions is the idea that a thread is a serial set of computations that can be paused, and then resumed at a later time [18], without changing the result. Hence it is possible to have many threads in flight on a single processor, by partially executing a thread, pausing, and then switching to other threads and the switching back. In GPGPU programs, there are usually many more threads (millions or billions) than processors (thousands).
warp	A group of 32 threads

5.2 The CUDA Programming Model

Since the limitations of programming on the GPU have a large impact on how an algorithm can be implemented in CUDA, or any GPGPU language, it is worthwhile to have an overview of the CUDA programming model and the anatomy of a typical GPGPU program. The motivation for these programming abstractions will be given in later sections. A program on the GPU is called a *kernel* in traditional literature. This can be confused with other usage of the term in this thesis, so will be referred to as a "*GPU kernel*" in this thesis.

As mentioned earlier, programming the GPU can be achieved using CUDA

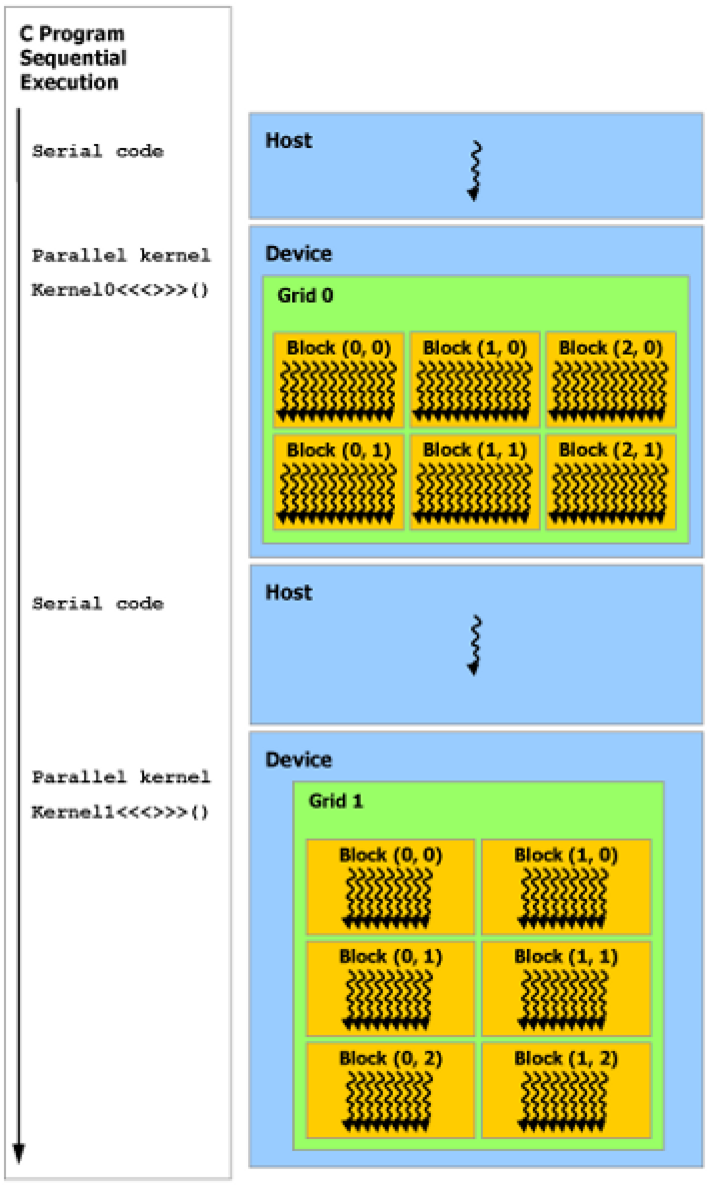

Figure 5.2.1: Diagram from [138] of a typical GPGPU program.

C++. Even though this programming language is based on C++, there are several important differences to programming a single core CPU program. This arises from the fact that the CPU and GPU are built for different purposes and therefore have different structures and properties. The GPU is far less flexible than a CPU. For example, it cannot run an operating system. The GPU requires a lot of intervention from the CPU in performing a computation.

In a typical GPGPU program, the data that will be used for the computation has to be initialized by the CPU. The data at the start of the program resides only on the CPU and needs to be transferred on to the GPU. This involves the explicit instruction of the GPU to set aside memory. The GPU will then allocate the memory and pass back an address to the CPU. Note that everything needs to be called from the CPU, so this address needs to be stored on the CPU in case it needs to be used later. The next thing to be done is to copy the data from the CPU to the GPU. This is done using several commands in the CUDA language, which instruct the CPU to copy the data from the CPU to the GPU at the address obtained earlier. This address is situated in what is known as *global memory*, a programming abstraction, which refers to memory which is accessible from both the GPU (referred to as the device) and the CPU (known as the host). The reason for these programming abstractions is probably to allow the programming language to be able to program for future GPUs. The global memory, at this time of writing refers to the DRAM situated on the graphics card. This DRAM is a temporary, relatively fast memory, where the data are stored in capacitors. It however is off-chip, which makes it slow in GPGPU terms.

A parallel program consists of many threads: sequential series of computations which can be paused and resumed without changing the result of the computations. These threads in a GPU program are not autonomous, because of hardware limitations, instead these threads are executed in groups of 32, known as *warps*, in which all threads in the warp must execute the same instructions as each other. "If" statements are dealt with by executing both cases of the statement, and discarding the result which does not apply for each thread (there are ways

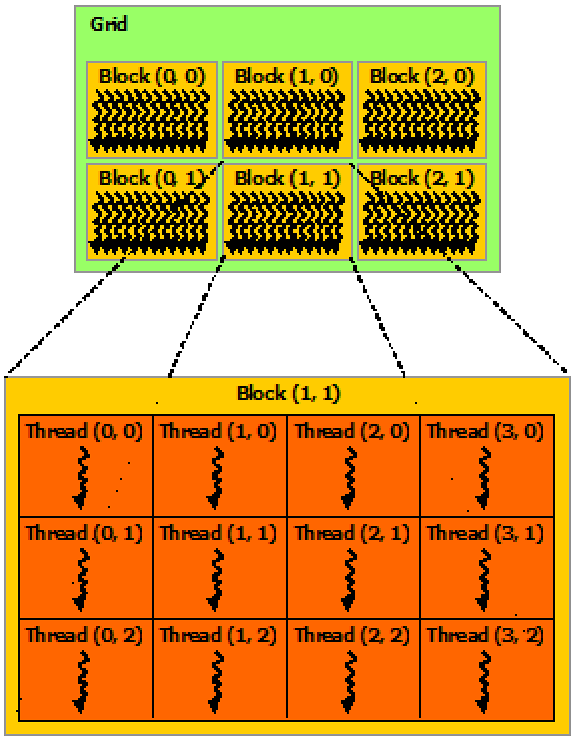

Figure 5.2.2: Diagram from [138] of CUDA thread model. A collection of threads, indexed by a coordinate, make up a block. The coordinate can be one, two or three dimensional. Several blocks, indexed by a coordinate, make up a grid. A CUDA kernel executes on a grid.

that the compiler and GPU can avoid this, which are beyond the scope of this chapter).

To make it easier to work with threads, most programming languages, including CUDA, group threads into *blocks* of threads (fig. 5.2.2). A block is a programming abstraction which consists of a specified number of threads. Several blocks make a *grid*. The blocks and grids are indexed by a 1,2 or 3-dimensional coordinate, which can be referred to in the programming language. Grid and block size are specified by the programmer in the code.

The GPU kernel can now be launched, by calling it from the CPU explicitly, by using commands in the CUDA language. The kernel is run, the same instructions being run in all threads in the grid. But suppose the programmer wants each thread to perform different instructions? This can be done by using an "if" statement, where different cases of the "if" statement refer to different thread indexes (for example, “if the thread x index is less than 1000, do ...”). This is why each thread has an index. Providing that the same branch of the “if” statement is taken for all the threads in each warp, there is no performance loss due to branching. If there is one or more threads that follows the other branch of the “if” statement, both branches of the if statement will have to be followed by the whole warp, with each thread only keeping the result that is needed. This is known as “branch divergence”, and is to be avoided in GPGPU programming.

During the GPU computation the CPU thread that launched the GPU kernel runs asynchronously, until it reaches a command given in the code to wait for the GPU to finish (called a barrier). Ones this barrier has been reached by the GPU, the CPU thread resumes. When the GPU has finished its computation, the programmer needs to indicate in the code for the CPU to copy the results back into the main RAM. The global memory has to be explicitly cleared via instructions in the code, otherwise the same memory will be available to the next kernel to be run. However, this can be desirable, as this keeps results from the last computation on the GPU, so that they do not need to be uploaded again. Memory copies from the CPU to the GPU take a relatively long time to perform.

These are the basic steps in a GPGPU program, and many of the languages used to program the GPU use these steps, although the terminology may be different for each language. There are simpler languages, but they do not yield the great performance gains typically obtained by more low-level languages.

These steps are not enough to gain a increase in performance. Often, code ported from a serial implementation will run slower than the GPU implementation. More detail will be given in further sections, but one of the main opportunities for increasing the performance of a program is to use what is called *shared memory*.

Shared memory (fig. 5.2.3) is memory that is on-chip and is approximately 100 times faster than the global memory. However, there are a lot of constraints to using this ultra-fast memory. Unlike the global memory, the shared memory can only be accessed from the threads on the device. This means that memory cannot be loaded directly onto shared memory. Instead, the data need to be loaded into the global memory, then commands need to be given in the GPU kernel code for threads to move the data on to the shared memory. Hence, shared memory should only be used for data that are to be used several times per thread. In addition,

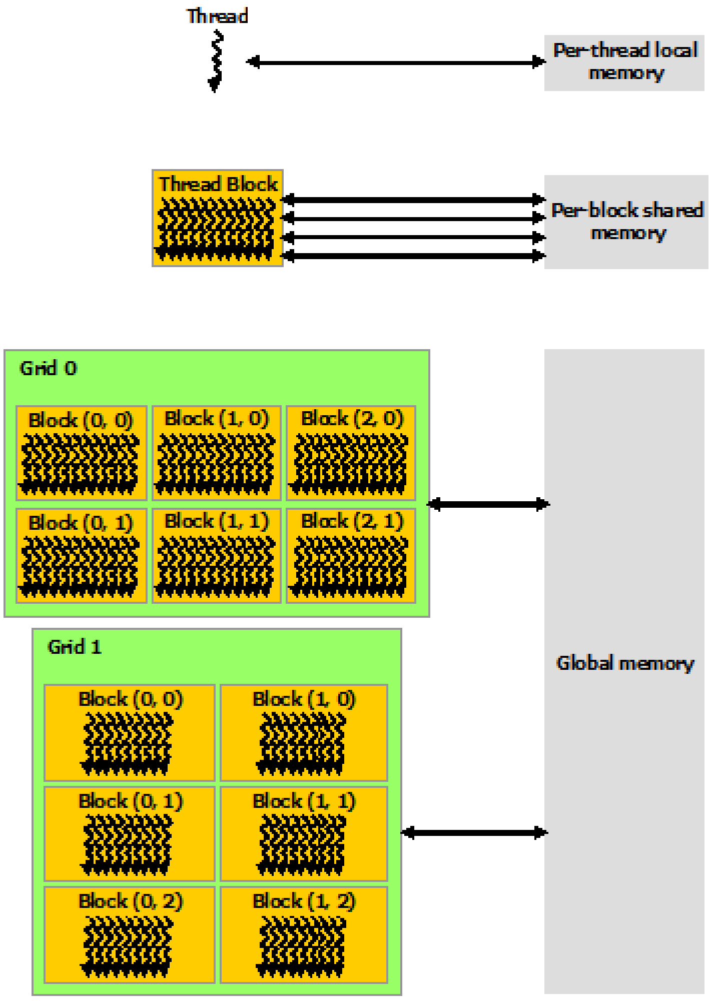

Figure 5.2.3: Diagram from [138] of CUDA memory model. Each thread can access thread local memory. Each thread in a block can also access block shared memory for the block that it is in. All threads in all blocks can access global memory. The bottom figure shows that if two kernels are running, one after another, or simultaneously (an advanced technique) on two different grids, all of their threads can access the same global memory.

as its name indicates, the memory is shared for each block, which means that threads in the same block use the same shared memory. This can be used to pass data from thread to thread.

Often many algorithms require data to be passed from thread to thread. To make sure that all the threads in the same block are at the same stage in the computation when they swap results, there is a command in the CUDA language to synchronize threads. This command creates a barrier in the code, where all the threads in the block must wait at until all the other threads in the block reach the barrier. At this point, data can be swapped between threads via the shared memory.

This concludes the brief description of the structure of a CUDA program and

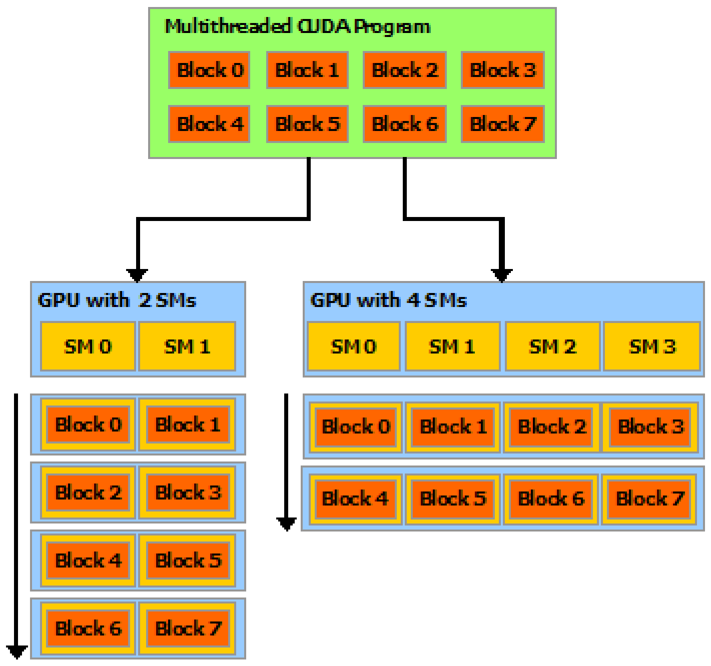

Figure 5.3.1: Diagram from [138] of CUDA how blocks in the grid are mapped to streaming multiprocessors during compilation.

the basic constraints in CUDA programming, but there is a lot more detail to be covered, as this only scratches the surface of a very complicated subject. The material covered here will yield a speed up of ten times upon serial code, but not the 300 times speed-ups which have been associated with GPGPU programming. Further detail on GPU program optimisation will be given in later sections.

5.3 GPU architecture and CUDA code scalability.

A basic understanding of GPU architecture is necessary to understand certain parts of this chapter. The GPU is built out of many streaming multiprocessors. Each of these streaming multiprocessors consist of 32 CUDA cores. One of the reasons why the code is written in blocks is that during compilation, these blocks are allocated by the compiler to streaming multiprocessors (see fig. 5.3.1). The reason for this is forward and backwards compatibility. Other models of GPU than the one used on the developer's computer may have more streaming multiprocessors, or fewer streaming multiprocessors. The block abstraction allows the same code to be compiled for different GPUs.

5.4 Challenges of GPGPU programming

Note from above that there are several challenges when programming on the GPU. There is limited autonomy between *threads* in the GPU – each thread in a warp needs to perform the same operations. The data need to be transferred to and from the GPU before and after the kernel has been completed, incurring overhead. The differences between the GPU and CPU create obstacles for the program. Programming on the GPU involves some low-level manipulation of memory to gain speed up. This is because there are several types of memory in the GPU. Global memory is slow but can be accessed from all threads. Local memory is very fast but is not shared between threads. Shared memory is shared between blocks of threads and is almost as fast as local memory (100x that of global memory). As mentioned in the previous section, attention must be given to memory allocation and usage to allow maximum performance. Indeed, the first implementation of a GPGPU program may run slower than a CPU implementation and careful performance tuning is necessary to gain performance which is superior to a CPU. Even carefully performance-tuned programs can run slowly when transferred to a computer with a different GPU from the GPU in the computer that they were developed with, as GPU technology has evolved rapidly over the last few years.

NVIDIA has released GPUs of each generation with a different architecture, each with different structures and performance properties. Each architecture is given a name (for example: "Tesla") and a number, denoting *compute capability*. Compute capability is a term often used interchangeably with architecture, but can indicate the capabilities of the GPU in a more specific way. For example, the first wave of CUDA GPUs were of the architecture "Tesla", which comprises GPUs of compute capability 1.0, 1.1, 1.2, 1.3. GPUs of compute capability $\geq$ 1.1 are able to perform integer atomics on 32-bit words in global memory, for example, whilst a GPU of compute capability 1.0 cannot do so. To date there have been Fermi, Kepler, Maxwell, Pascal, and Volta architectures. The differences between the architectures are too numerous to list here. For a full list of the differences between GPUs of different architecture, see [138]. The Kepler architecture has been used for the runs in this thesis, which was the latest architecture at the start of the project. A lot of the additions and advances in GPU technology have since been oriented towards machine learning (such as low-precision computation), which has been one of the major uses of GPGPU programming. This thesis will detail the methods used to allow the program to self-tune to whatever architecture of GPU is running on the computer.

As with all parallel programming, there is the issue of synchronization. Threads throughout the device can only be synchronised from the host (i.e. the CPU). Threads within each block can be synchronised from the GPU. Hence, there is an incentive to partition blocks such that the computation for each block is independent of other blocks.

5.5 Parallel Programming Patterns and Parallel Programming Libraries

This section begins with a review of two of the most important parallel programming patterns [120]: reduction and map. These will be used to show how apparently trivial algorithms can become complicated to implement using GPGPU, despite the speed benefits. This section will conclude with a discussion of how parallel programming libraries aid the parallelisation and tuning of algorithms that fit common parallel programming patterns.

Many programs follow programming patterns (also known as algorithmic skeletons)[120]. The first is “*reduction*”.

Example 21 (Parallel Reduction)**.** Parallel reduction, a parallel generalisation of serial reduction, is an example of a commonly used programming pattern which needs to be parallelised.

Definition 22 (Reduction)**.** Suppose we have a binary commutative and associative operator $\oplus : X^2 \to X$. Let $C = \{f | f : X^2 \to X\}$. Then a reduction (a term attributed to [1]) reduce : $X^n \times C \to X$ of an array $V \in X^n, V = (v_1, v_2, \ldots, v_n)$ is a function defined by (using notation similar to [22, 124]):

$$\text{reduce}(V, \oplus) = v_1 \oplus v_2 \oplus \ldots \oplus v_n$$

This is also known as a *fold* in functional programming (there are left and right folds, which are outside the scope of this example, and for associative operations a left fold is a right fold). The name "reduction" is thought to originate from the Lisp programming language [119, 167]. This function is known as `Reduce(func, list, initval)` in R, `Fold[func, initval, list]` in Mathematica and `std::accumulate(begin, end, initval, func)` in C++.

This will be used to combine the different parts of the log-likelihood into a single sum.

Corollary 23 (Summation is a reduction)**.** *Observe that:*

$$\sum_{i=1}^{n} v_i = \textit{reduce}(V, +)$$

Corollary 24 (Product is a reduction)**.** *Observe that:*

$$\prod_{i=1}^{n} v_i = \textit{reduce}(V, \cdot)$$

The most common algorithm used to implement this on a single processor is as follows:

1. Create a variable called "`sum`". Set this to 0.0
2. For $i = 1, 2, \ldots, n$, add v_i to `sum`
3. Return `sum`

This algorithm is not an efficient implementation a reduction on a parallel processor, because there is no code that performs any parallel operations. Therefore, there is no benefit from using a parallel processor. Another algorithm is needed, which processes several items in parallel.

A commonly used algorithm, to implement a reduction on the GPU is the cascading reduce, where the elements of the first half of the array are combined with the second half, the second half of the array is deleted, and this process is repeated until there is a single element remaining, which is the result of the reduction. This algorithm was created so that thousands of additions or multiplications (or any other reduction operation) can be performed simultaneously on the GPU, which has thousands of processors. For example, to parallelise the reduction for $n = 2^k$ and $\oplus$ commutative, the associative and commutative property of $\oplus$ is used. For example, for $n = 8$:

$$\begin{aligned}
\text{reduce}(V, \oplus) &= v_1 \oplus v_2 \oplus v_3 \oplus v_4 \oplus v_5 \oplus v_6 \oplus v_7 \oplus v_8 \\
&= (v_1 \oplus v_5) \oplus (v_2 \oplus v_6) \oplus (v_3 \oplus v_7) \oplus (v_4 \oplus v_8) \\
&= ((v_1 \oplus v_5) \oplus (v_3 \oplus v_7)) \oplus ((v_2 \oplus v_6) \oplus (v_4 \oplus v_8))
\end{aligned}$$

The reduction is done in steps, where the elements of the first half of the array are combined element-wise with the second half using $\oplus$, the second half of the array is deleted, and this process is repeated until there is a single element remaining, which is the result of the reduction.

That is, at step 0 we have:

$$V_0 = (v_1, \ldots, v_n)$$

and at step 1 we have:

$$V_1 = ((v_1 \oplus v_{n/2+1}), (v_2 \oplus v_{n/2+2}), \ldots, (v_{n/2} \oplus v_n))$$

and so forth until we have

$$V_k = (v_1 \oplus v_2 \oplus \ldots \oplus v_n)$$

V_i is stored in global memory and at each step all the GPU processors are synchronised by the CPU. This is an example of how an apparently simple algorithm sometimes becomes complicated when forming a parallel algorithm. In addition, there are more complicated algorithms for the more general case which take $n \in \mathbb{N}$ and $\oplus$ non-commutative. There are also algorithms with other performance characteristics which may be desirable in various circumstances. However, many of the simple reduction algorithms have been implemented in ready-to-use modules in CUDA CUB, which contains several implementations of reduction. The efficient implementation of fast reduction on the GPU is an active research topic in computer science (for example, [81, 117, 86]).

Example 25. An example of another commonly used algorithm in our code is: $\mathrm{map} : X^n \times (X \to X) \to X^n$

Definition 26 (Map)**.** $\mathrm{map} : X^n \times (X \to X) \to X^n$ is a higher order function defined by:

$$\mathrm{map}(V, f) = (f(v_1), f(v_2), \dots, f(v_n))$$

This can be parallelised in a straightforward manner by loading v_i into processor i, performing f upon it and writing the result to global memory. If the number of elements in V exceeds the number of processor, each processor will process several elements. There are several variants of this algorithm which use different types of memory, for different performance properties, and the granularity (how many elements are processed by each processor) can be tuned for performance, due to latency or throughput issues with the calculation of f. Therefore, it is important to note that it is difficult to predict which algorithms are easy to parallelize, as relatively similar algorithms can lead to very different implementations.

However, the complexity of implementing such algorithms like *map* and *reduce*, which are sometimes called *primitives*, has led to the creation of libraries such as CUDA CUB, which is used in this thesis. CUDA CUB has implemented several GPU primitives, such as map and reduce, using CUDA and template C++, such that on compilation, the compiler detects what GPU is fitted in the computer and selects the best algorithm automatically. Through the use of these libraries, the impact of different GPU architectures is minimized by using an algorithm that can be expressed in terms of these primitives. However, such libraries are not sufficiently flexible for everything to be expressible in terms of library functions, and not all programs, including the programs used in this thesis, fit patterns, and require extensive hand coding and tuning of certain parts of the algorithm.

5.6 GPGPU program optimisation principles and patterns

The total run time of a serial program is the sum of the execution times of all the individual instructions in the program. For a parallel program when many instructions are run simultaneously this is not the case. Instead the performance of a parallel program is dependent on many complicated and interacting factors (for example, see [186, 185, 187, 81]. Execution time is dependent upon the latency and throughput of each item i.e. memory transactions, instructions etc. This is referred to by some computer scientists as Little's law [113]. The latency of each item is the difference between the start and end time of each item. The throughput is the rate at which items can be processed, usually measured in items per clock cycle.

The latency on the GPU is mostly affected by memory loads and stores and is usually hidden by keeping as many threads in flight as possible (GPU thread switching incurs relatively little performance loss). The GPU is a throughput focused processor, but branch divergence (i.e. conditional statements) can halve the computational throughput since groups of 32 threads (warps) need follow exactly the same commands. In addition, occupancy is affected by block size which then affects the number of threads in flight. There is a limited amount of shared memory and registers available, so only a limited number of threads can run at the same time on the same SM (Streaming Multiprocessor, see Section 5.3) and hence only a certain number of thread blocks.

Hence, optimisation of GPU code is a difficult task. Nevertheless, in [170] in a review of highly optimised CUDA code in the existing literature, found that there were seven major techniques of optimising performance. These techniques are not used in isolation; one technique tends to reveal an opportunity to use another in the list. In addition, the list is not exhaustive; there are many low-level and architecture-specific optimisation techniques which can be used. However, the list gives a high-level view of the general techniques in improving parallel program performance.

5.6.1 Data layout transformation

This technique involves a transformation of the data layout to optimise reads and writes from memory. Global memory is read in "chunks", thus, to optimise the use of memory bandwidth, adjacent threads should read adjacent memory locations, in order to minimise the number of memory transactions. This is called *burst utilisation [115].* There are many transformations that can be performed (some of which are very complicated and beyond the scope of this thesis). The most common transformation is transforming an array of structures into a structure of arrays (see fig. 5.6.1), as the latter has better memory performance, since the reads are from memory locations adjacent to each other.

5.6.2 Scatter to Gather Conversion

A *scatter* operation is defined as an operation in which each thread reads an element of data (whose location is known) and produces many results (which may or may not be statistically random in location), which are written to different locations in the output vector. A *gather* operation is the opposite, in which each thread reads many inputs (which may or may not be statistically random in location) and writes to a single output location per thread (whose location is known).

Gather operations are preferable for parallel programming because they avoid many conflicting writes to the same location. When conflicting writes are made to the same location, two or more threads may write to the same location at once.

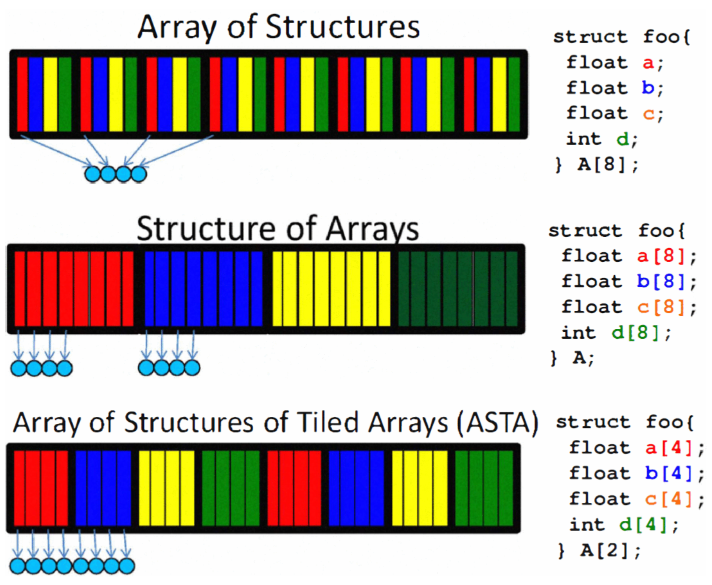

Figure 5.6.1: Diagram from [170] of how data layout transformation can allow efficient use of memory bandwidth. In the top figure, each thread represented by the blue circles processes every fourth element in the data. This is because the data are defined in the code on the right and an array of structures. Transforming the data to be represent as a structure of arrays, represented by the middle diagram now causes the threads to access adjacent entries in memory, allowing more burst utilization. The bottom figure shows a more complex data transformation that allows all threads to access entries in the memory which are next to each other.

This creates what is known as a memory race. To avoid each thread from making conflicting writes, quite often *atomic* [148] operations (operations that appear to each thread as an indivisible and uninterruptible single operation) are used. An atomic memory operation is a memory operation which completes in one step relative to other threads. Hence, when a thread is performing an atomic memory write, the memory location is never visible as half complete to other threads, hence cannot be modified by other threads while the write is in progress. Using these produces a significant loss in performance, because it essentially makes the memory writes serial [115].

A *scatter-to-gather* operation is a transformation of the program or kernel which allows the computation to be changed from a scatter operation to a gather operation (see fig. 5.6.2). By converting the scatter to a gather the conflicting writes can be eliminated without resorting to using costly atomic operations.

5.6.3 Tiling

Shared memory is a lot faster than Global memory, so if values are repeatedly being loaded or stored, the data for those threads can be loaded from the Global

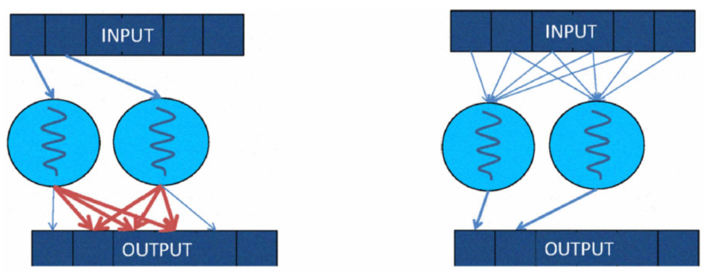

Figure 5.6.2: Diagram from [170] of the scatter to gather parallel optimisation pattern. In this pattern, a scatter, in the left figure, an operation which reads one input location per thread, but writes to several locations causes several threads to write to the same location at once. These conflicting write operation are denoted by the red arrows, and the non-conflicting memory operations are denoted by the blue arrows. The right figure shows what happens when this scatter operation is converted to a gather operation. The gather operation reads several memory locations per thread, but only writes to a single output location per thread. By using this technique the conflicting writes have been eliminated without resorting to using costly atomic operations.

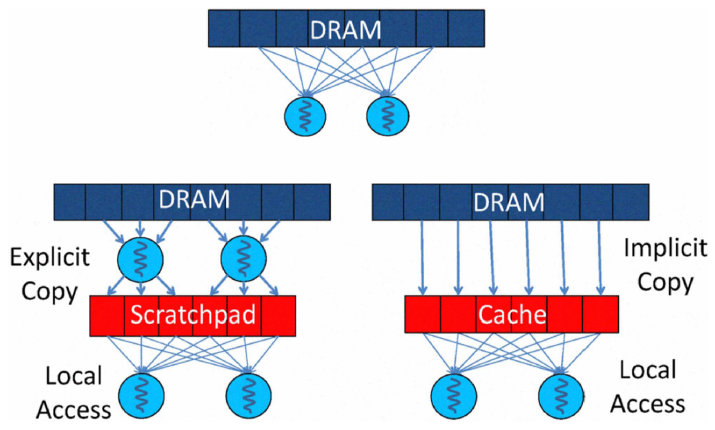

Figure 5.6.3: Diagram from [170] of tiling. In the top figure, the unoptimised code reads from global memory. Tiling, shown on the bottom left moves this data into shared memory, using the shared memory as a scratchpad. This shared memory is approximately a hundred times faster than global memory, allowing threads which reuse the data several times to avoid reading from global memory several times. This is analogous to the tiling pattern for CPU parallel program optimisation, shown on the bottom right, except that the copying needs to be done explicitly in the GPU code, instead of implicit copying done on the CPU code, and often not all of the data can be copied into limited shared memory on the GPU, so copying needs to be selective.

memory onto shared memory and accessed from there. This allows memory which is being repeatedly used between several different threads to be quickly accessed (see fig. 5.6.3). This only gives a speed-up if memory if being repeatedly reused, as all data must first be transferred from the CPU to GPU Global memory and cannot be transferred directly into Shared memory. Also, increasing shared memory per block lowers the number of blocks that can be run on each streaming multiprocessor as memory is a limited resource, leading to lower occupancy.

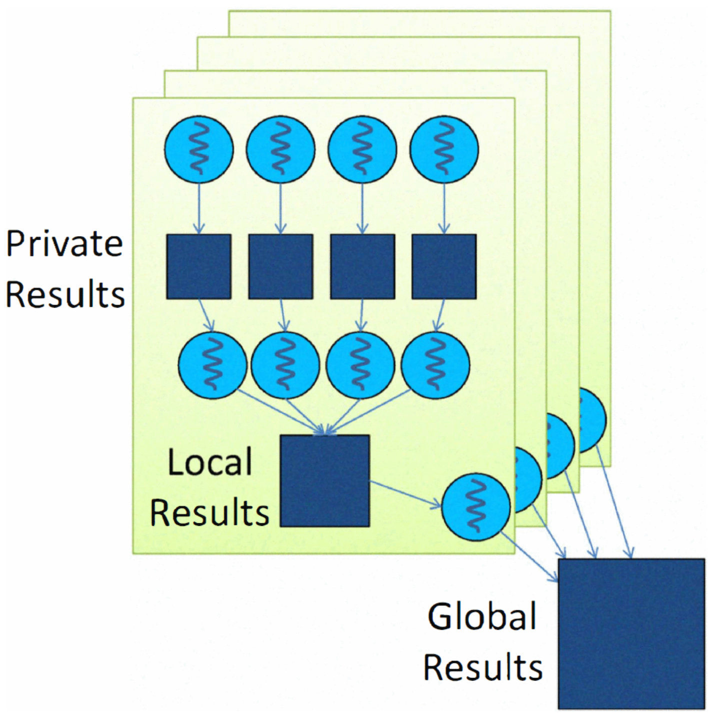

Figure 5.6.4: Diagram from [170] of privatization. The optimised code is written such that each thread produces its own local results, only merging them at the end of each block"s computation into a global result. This avoids writing to the same output location repeatedly.

5.6.4 Privatisation

Privatisation is the opposite of tiling. In this case, a private copy is made at the thread or block level of data that is common between several threads or blocks, so that threads and blocks can operate independently of each other. At the end of each thread/block's computation the private results are merged into a single Global result (see fig. 5.6.4). The avoids several threads writing to the same output memory location repeatedly, as opposed to tiling, which avoids repeated reads from the same input location.

5.6.5 Binning

It is sometimes unclear what input elements are taken by each thread. Creating a data structure that maps output elements to small sets of possible input elements is called *binning* (see fig. 5.6.5). This makes it possible to then use the scatter-to-gather conversion technique.

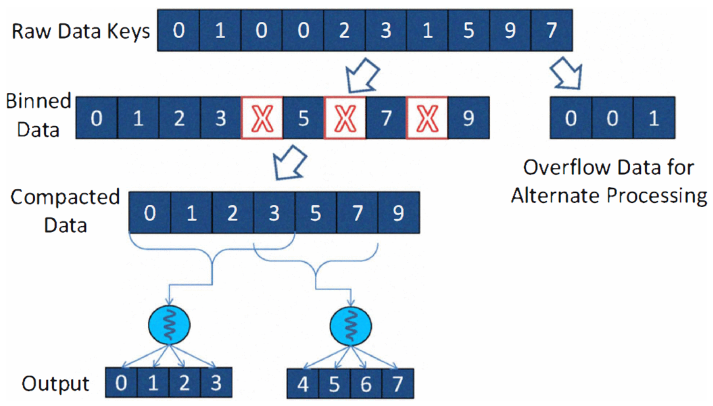

Figure 5.6.5: Diagram from [170] of binning (top), compaction (middle), and regularization (bottom). It is common to use several of the seven GPU optimisation patterns in the same program.

5.6.6 Compaction

Often a computation taking n inputs and m outputs is parallelised using $\frac{n}{k}$ threads where k is the number of inputs per thread. If a large number of threads do not produce any output, many of the threads will do very little processing, especially if a single element is processed per thread (in which there is likely to be a conditional statement determining whether that thread produces an output or not). Eliminating entries which do not affect the output is known as compaction (see fig. 5.6.5).

5.6.7 Regularisation

This involves load balancing threads, such that a minimal number of threads are idle (see fig. 5.6.5).

5.7 Implementation of GPU accelerated RJMCMC

5.7.1 Identifying the bottleneck

The RJMCMC algorithm (see Section 4.2.4.1) is highly serial. There are two routes to accelerating RJMCMC on the GPU [16]. One option would be to run parallel chains on different processors, but this would require several million concurrent chains for the GPU to be sufficiently occupied, in which case burn-in of the chain would be difficult to assess, and large numbers of iterations would be discarded as burn-in. Another approach, which is the approach taken in this thesis, would to be to run a single chain, and reduce the time for each iteration.

Consider each iteration of a single chain. Each step in the iteration is serial dependent on the previous. However, there is a clear bottleneck in the calculation for the likelihood. As the amount of data increases the time taken to evaluate the posterior increases greatly.

In [110], MCMC is parallelized for a model for cancer data in the US. All of the steps in the MCMC are performed on the CPU, including the Metropolis step for updating one parameter and a Gibbs sampler step for another parameter. However, the full conditional distribution is derived analytically for several of the parameters, and these parameters are updated as a block in one Gibbs sampler step, the calculation being performed on the GPU. This uses an algorithm to generate many Gamma distributed random numbers in parallel. The authors of [176] also use the GPU to update large blocks of variables simultaneously.

In [181], an approach is proposed in which data augmentation is used to formulate GPU algorithms, in which the augmented data are updated in a single Gibbs step. If the model is exchangeable, that is, for each observation y_i there exists a latent z_i such that

$$\pi(y_i|\theta) = \int \pi(y_i|z_i)\pi(z_i|\theta)\, dz_i$$

then z can be updated in a single Gibbs (or Gibbs within Metropolis) step. This is demonstrated for horseshoe probit regression.

In these papers, Gibbs samplers (or Gibbs within Metropolis) are used to update large amounts of variables in a single parallel operation. A possible candidate for such a strategy within epidemic models is the update of the augmented data. However, this greatly increases the complexity of the code of the implementation, when already GPGPU code is very low-level and complex. If an analytic form of the full conditional can be found, a separate code path would need to be implemented for the block update of the data, and this would need to be tested and updated, which is labour intensive, considering the work for this thesis required a lot experimentation and modification. If the full conditional cannot be analytically found it will be challenging to construct a block update algorithm with sufficient acceptance rate. However, the approach of updating large blocks of variables appears promising and further investigation is needed.

A possible idea for updating the unobserved data when using MCMC for fitting an epidemic model is to use a non-centred parametrisation [107]. For example, the residuals in section 3.3.2 are independent of the model parameters and each other, and are uniformly distributed allowing them to be updated in parallel in one step. However, parametrising the model in this way will require re-expressing the likelihood. It is not clear, as in most GPU programs, whether this method gives a speed gain over the method described here.

In the field of epidemic modelling there has been little work on creating GP-GPU implementations for data augmented MCMC.

In RJMCMC, the slowest step is the calculation of the acceptance ratio, of which the calculation of the likelihood at the proposed parameter and augmented data values is a large part. A large speed gain can be obtained by accelerating the calculation of the likelihood. In addition, unlike the papers featured here, this thesis is an investigation into the effectiveness of model comparison methods, several of which require maximum-likelihood estimation as well as MCMC. These methods also benefit from the acceleration of the likelihood, resulting in large speed gains.

5.8 Previous work on computational efficiency for epidemic models

In [29] the authors detail attempts to speed up MCMC on an SEIR model for aphids on sugar cane plants in a plantation on the island of Guadeloupe. The authors used the following approaches:

1. Pre-computation and storage of the distance between each host. In addition, they compute a simplified version of the acceptance ratio for MCMC, can-

celling terms that appear both on the denominator and the numerator. This is a common approach used by most MCMC implementations.

2. The algorithm was parallelised. Since "snapshot" data are used, when infections are known to have occurred in different time intervals, proposals which violate this ordering are instantly rejected, and infection times are updated in parallel when they occur in different intervals. The transmission kernel is also pre-computed and stored.

3. Since an isotropic transmission kernel is symmetric in i and j, the number of calculations can then be halved. Finally they truncated the transmission kernel, assuming there are no secondary infections beyond a certain distance. They also used a discrete-time approximation of the model.

The basic algorithm (with no optimisations) took 232.33 seconds for 100 iterations. The parallel implementation took 92.81 seconds for 100 iterations. The implementation with all the optimisations apart from the truncation took 25.12 seconds for 100 iterations. The implementation with truncation of the spatial kernel took 5.03 seconds for 100 iterations.

Two discrete-time approximations were used, with the infection events only possible at a discrete set of time points. The unoptimised discrete-time approximation implementation toon 901.23 seconds for 100 iterations. When the spatial kernel was truncated as before this implementation took 35.95 seconds for 100 iterations.

The CUDA blog features an interview with Chris Jewell of Lancaster University [133]. In the interview, published in 2015, he details how using GPGPU with CUDA has benefited his work, and the challenges and best practices that he found in his programming experience. As of the time of writing, no paper has been published detailing his methods, but a code repository is available online on GitHub [93]. According to [93] the algorithm focused on offloading the calculation of the likelihood to the GPU, and yielded substantial speed-up on the 2001 FMD data set, which involved 188361 hosts (in this case farms are considered to be hosts).

The model fitted to the data consists of four states: susceptible S, infected I, notified N, removed R. Farms which move onto the next stage of infection cannot move backwards and recover. Farms which are in the notified state and the infected state can infect animals in the susceptible state. There is a period in which animals are infected but not infective, which is four days. Let $\lambda(j,t)$ be the infectious pressure on susceptible j at time t, then:

$$\lambda(j,t) = \epsilon(t) + \sum_{i \in I(t)} \beta_{ij} h(t_j^I - t_i^I) + \sum_{i \in N(t)} \beta_{ij}^* h(t_j^I - t_i^I)$$

where

$$h(t) = \begin{cases} 0 & t < 4 \text{ days} \\ 1 & \text{otherwise.} \end{cases}$$

The movement ban is modelled by altering the primary infection rate, so that

$$\epsilon(t) = \begin{cases} \epsilon_1 & t < \text{movement ban} \\ \epsilon_1 \epsilon_2 & \text{otherwise.} \end{cases}$$

The secondary infection rate consists of a susceptibility term, an infectivity term and a distance-dependency term.

For $i \in I(t), j \in S(t)$:

$$\begin{aligned} \beta_{ij} &= \gamma_1 \beta_i^{(I)} \beta_j^{(S)} \beta_{ij}^{(D)} \\ \beta_{ij}^* &= \gamma_2 \beta_{ij} \end{aligned}$$

where c_i, s_i, p_i are the numbers of cattle, pigs and sheep at site i, $\bar{c}, \bar{s}, \bar{p}$ are the mean numbers of cattle, pigs and sheep at each site. Here the susceptibility $\beta^{(S)}$, infectivity $\beta^{(I)}$ and distance dependency terms $\beta^{(D)}$ are:

$$\begin{aligned}
\beta_i^{(I)} &= \left(\frac{c_i}{\bar{c}}\right)^{\psi_1} + \xi_2 \left(\frac{p_i}{\bar{p}}\right)^{\psi_2} + \xi_3 \left(\frac{s_i}{\bar{p}}\right)^{\psi_3} \\
\beta_j^{(S)} &= \left(\frac{c_j}{\bar{c}}\right)^{\phi_1} + \zeta_2 \left(\frac{p_j}{\bar{p}}\right)^{\phi_2} + \zeta_3 \left(\frac{s_j}{\bar{p}}\right)^{\phi_3} \\
\beta_{ij}^{(D)} &= \frac{\delta}{(\delta^2 + \rho_{ij}^2)^{\omega}}
\end{aligned}$$

Reading through the source code was challenging, as it is written as part of a larger application, which involves data reading, output and visualization. Nevertheless, the relevant files were identified. A brief read through of the approximately 3000 lines of code required to implement calculation of the likelihood on the GPU, several notable features of the implementation were noted. The GPGPU code is written in CUDA, with a total of thirty different kernels for maintaining and uploading the data to the GPU.

Part of the reason why many kernels are needed to maintain the data is due to a key approximation made on the calculation of the transmission kernel. Outside a radius of 25 kilometres the transmission kernel is assumed to be zero. Unlike the algorithm in this thesis, instead of caching the transmission kernel, the distance matrix is cached. This distance matrix would be too large to cache, for the data set consists of 188361 farms, each of which is considered a host. Instead, the distances between each host are evaluated and the entries for those pairs of hosts which are further than the truncation radius are discarded. This in effect changes the spatial kernel to:

$$\beta_{ij}^{(D)} = \begin{cases} \frac{\delta}{(\delta^2+\rho_{ij}^2)^{\omega}} & \rho_{ij} < 25\text{km} \\ 0 & \text{Otherwise} \end{cases}$$

This allows the distance matrix to be stored in a space-saving format known as a sparse matrix. This stores a $m \times n$ matrix M as three arrays (`A`, `IA`, `JA`). `A` stores all the non-zero entries of M. Let m_{ij} be a non-zero entry of M. The zero entries of m_{ij} are not stored. The ith element of `IA` holds the index of the first element of A that is in row i (in programming terms, called the row pointer). The array JA stores the column indexes from the original matrix: the kth element of JA holds the column number (in the original matrix M) of the kth element of A. For example, if the value of $m_{ij} = 1$ were stored at A[k], then JA[k] would be j, and A[k] = 1.

Jewell's algorithm partitions the calculations in a manner that differs from our approach, with the intermediate calculations stored on the GPU between calculations and the code written. With the FMD model, the force of infection between an infected individual and a susceptible individual, is comprised of a term representing the susceptibility of the host (which depends on the mix of animals present at the site), a term representing infectivity of the infector, and a distance dependency term, which is the spatial kernel. The data stored on the GPU are arrays of the susceptibility and infectivity of each host premises, and the distance matrix, and the observed data and a vector of the unobserved infections. These are only updated when they are changed. Jewell's code has different updated routines that are run depending on the circumstance: recalculate the full likelihood, recalculate the likelihood but avoid power calculations by using the values $\left(\frac{c_i}{\bar{c}}\right)^{\psi_1}$, $\left(\frac{p_i}{\bar{p}}\right)^{\psi_2}$, $\left(\frac{s_i}{\bar{p}}\right)^{\psi_3}$, $\left(\frac{c_j}{\bar{c}}\right)^{\phi_1}$, $\left(\frac{p_j}{\bar{p}}\right)^{\phi_2}$ and $\left(\frac{s_j}{\bar{p}}\right)^{\phi_3}$ which are loaded from stored memory. The latter is used when the parameters $\psi_1, \psi_2, \psi_3, \phi_1, \phi_2$ and ϕ_3 have not changed between successive evaluations of the likelihood. This is far more complicated than the implementation in this thesis, which has only one update routine. Nevertheless the implementation presented in this thesis appears to offer a considerable gain in speed compared to Jewell's implementation. Depending on which routine is called, different GPU kernels are called. These include a kernel to update the "product" (analogous to S_{1i}), and one to update the "integral" (analogous to S_{2i}). In addition, there are GPU kernels to calculate the "infectivity" $\beta_i^{(S)}$, "susceptibility" $\beta_i^{(S)}$, and the primary infection integral $\int_0^t \epsilon(x)dx$. These are computed in a 1-dimensional grid (as opposed to a 2-dimensional grid in this thesis), and reduced using a library called CUDPP [82, 83, 162, 163, 183]. These stages are implemented as map-reduce patterns (although not explicitly named as such in the code, nor in the blog post).

This appears to be an adequate strategy for large data-sets (hundreds of thousands of hosts). However, there are several features which make this approach unsuitable for medium to moderately large data-sets (thousands of hosts). In fact, there appears to be work ongoing to make the code switch to the CPU automatically in this circumstance [156]. There are several features that lower the efficiency of the implementation. The first is the use of sparse matrices. The data layout of sparse matrices hinders the efficient accessing of memory, since the matrix is stored in three arrays instead of one, the index array and row pointer array needs to be accessed as well as the data array when performing calculations. Such global memory operations are costly on the GPU. Because of the truncation and the use of sparse matrices, memory accesses to the observed data (not the accesses to the sparse matrix data, but any other data that need to be accessed as a result of the sparse matrix data) tend to be uncoalesced, leading to inefficient use of bandwidth. The implementation in this thesis uses dense matrices, which lower the amount of memory accesses, and increase the likelihood of coalesced memory accesses. In addition, there are no approximations involved (such as the 25km spatial kernel truncation mentioned above) therefore calculation of the likelihood is exact.

In addition, the strategy of recalculating the likelihood in separate parts leads to several inefficiencies: first of all, this has led to separate parts of the likelihood being calculated in different GPU kernels. This serialises a lot of the calculation, exploiting less of the possibilities of parallelism. In addition, each kernel launch incurs a performance penalty. A kernel takes a set amount of time to launch and complete regardless of what it does. Launching too many kernels serially hinders performance greatly. In this thesis, the overhead of kernel launch was found to be a substantial proportion of the run time of a kernel. A possibility would be to parallelize kernel launches, leading to further complexity in the code. This may not be possible as there may be not enough resources on the GPU to run several of the kernels at once. The algorithm in this thesis is designed to calculate the “product”, “integral”, and “background integral” in one kernel, instead of three, and does not serialize these tasks. Instead of running several kernels simultaneously, the single internal reuses memory, such that all stages of the map calculation can be performed in a single kernel, and will not encounter the resource limitation caused by parallel kernel launches. Performing the map operation in one kernel instead of several map operations in several kernels incurs minimal overhead.

It may seem paradoxical that performing fewer calculations may hinder performance, but GPU performance is often unintuitive, particularly because memory operations are slow relative to calculation, to the extent that it might sometimes be faster to recalculate a value than to load from memory. In addition, it is not so much the number of operations but rather the level of parallelism exposed: having more threads in flight can hide the latency of costly memory operations. However, having too many threads in flight may incur a performance cost. Finding the optimal balance is a non-trivial task.

In the code by Jewell, the calculations appear to be performed on a 1-dimensional grid, whilst the algorithm here uses a two dimensional grid, which allows better control of granularity and generates more threads, allowing the GPU to hide memory latency by switching thread. A possible motivation for this is that the map-reduce pattern in this calculation may be seen is analogous to sparse matrix multiplication. However, the difference between the calculations here and the matrix multiplication task is that a lot more memory accesses are needed during the map phase (the multiplication of individual elements). The latency needs to be hidden by keeping more threads in flight, so that the GPU can hide this latency by switching threads whilst the memory load is in progress. the implementation here uses a 2-dimensional grid for this purpose.

In addition, the calculations in Jewell’s code are float or 32-bit precision. This means each decimal number in the program is represented by a 32-bit binary code. In the programs used for this thesis the calculations are in double precision, which means that in each calculation the numbers or represent by a 64 bit binary code. Thus, the calculations are rounded to a larger number of significant figures, allowing more precise results. Finally the last difference between the code in this thesis and Jewell's code is that the code in this thesis auto-tunes itself to the GPU present on each computer that it is run on. Macros in the code select the most appropriate code path depending on the model of the GPU, and libraries which are capable of auto-tuning are used, allowing the algorithm to be optimised for whatever GPGPU the algorithm is run on. Jewell's code does not have this capability, although development is in progress to shift calculation from the GPU back to the CPU when GPU performance is poor. .

The PhD thesis [28], focuses on implementing MCMC for compartmental spatio-temporal epidemic models. The thesis is focused at creating an easy-to-use library for R which is programmed in C++. The model used was an SEIRS model, in which the "hosts" were sites comprising many individuals. Instead of using a spatial kernel, the author used a distance matrix to model the interaction of different sites over distance. The code was implemented in OpenCL, a library for programming on the GPU. The reason for using OpenCL instead of CUDA was that OpenCL code can run on GPUs produced by any manufacturer, whilst CUDA code is limited to GPUs produced by NVIDIA. This flexibility is at the cost of ease of use – OpenCL code is very verbose, requiring many more lines of code than CUDA. The author of the thesis found that "*As of this writing, the OpenCL features of libSpatialSEIR are not being used for any of our own data analyses or simulations. While much effort was devoted to developing the tools required to perform the aforementioned full conditional distribution evaluations in parallel, doing so has not yet proved practical on the wide variety of CPU and GPU hardware available to us. The problem size for which parallelization the MCMC sampling algorithms described in the rest of this chapter begins to pay dividends is still impractical;*". The author used two methods to speed up the calculations on the GPU. Computation of the full conditionals required computations of several sums, in which each component of each sum was computed on the GPU and summed. The author remarked that this requires data transfer, which either implies that the data were not stored on the

GPU between iterations or the summation of the components was performed on the CPU, or that it might be one of the constraints of using OpenCL. In any case, the data transfer issues are minimized here, as data are only refreshed on the GPU when needed and stored between iterations, and only one number is transferred back at the end of the computation. The second performance optimisation by the author was to use the clBLAS library, which is a library contains algorithms for matrix operations. This library does not automatically adapt to whatever GPU is on the system, unlike the library used in this thesis.

In summary, the nature of GPU programming requires hand-tuned code to run effectively and outperform the CPU. There are so many different data-set sizes, model structures, algorithms used in statistical epidemiology that an implementation that would be suitable for the majority of uses would be prohibitively complicated, and difficult to debug and maintain. GPGPU is a powerful tool which allows the acceleration of computer code without the communication costs of cluster computers, and researchers in many areas of computational statistics may benefit from becoming acquainted with GPGPU programming.

5.9 Preliminary derivation

In order to parallelize the likelihood on the GPU, the likelihood needs to be expressed in a way that lends itself to easy implementation on the GPU. As with any computer program, the implementation of the calculation as an algorithm involves a lot of rearrangement of the formulae involved, such that it may be not obvious that the algorithm finally derived calculates the desired formulae. In this section, the derivation of the formulae used in the algorithm are given. Recall, as in Equation 2.2.5 on page 18:

$$
\begin{aligned}
L(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) \quad \propto \quad & \left(\prod_{i;t_E^{(i)}\le T} \exp\left[-\int_0^{t_E^{(i)}} C(x_i,t)\,dt\right]\cdot C(x_i,t_E^{(i)})\right)\cdot \\
& \left(\prod_{i;t_E^{(i)}>T} \exp\left[-\int_0^{T} C(x_i,t)\,dt\right]\right)\cdot \\
& \left(\prod_{i;t_I^{(i)}\le T} f_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right)\cdot \\
& \left(\prod_{i;t_I^{(i)}>T} \left(1-F_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right)\right)\cdot \\
& \left(\prod_{i;t_R^{(i)}\le T} f_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right)\cdot \\
& \left(\prod_{i;t_R^{(i)}>T} \left(1-F_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right)\right)
\end{aligned}
$$

As stated earlier, the log-likelihood is often calculated because the likelihood produces very small values.

$$
\begin{aligned}
l(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) \quad \propto \quad & \left(\sum_{i;t_E^{(i)}\le T} -\int_0^{t_E^{(i)}} C(x_i,t)\,dt + \sum_{i;t_E^{(i)}\le T} \log C(x_i,t_E^{(i)})\right)+ \\
& \sum_{i;t_E^{(i)}>T} -\int_0^{T} C(x_i,t)\,dt + \\
& \sum_{i;t_I^{(i)}\le T} \log\left(f_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right) + \\
& \sum_{i;t_I^{(i)}>T} \log\left(1-F_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right) + \\
& \sum_{i;t_R^{(i)}\le T} \log\left(f_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right) \\
& \sum_{i;t_R^{(i)}>T} \log\left(1-F_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right)
\end{aligned}
$$

Merging $\sum_{i;t_E^{(i)}\le T} -\int_0^{t_E^{(i)}} C(x_i,t)\,dt$ and $\sum_{i;t_E^{(i)}>T} -\int_0^{T} C(x_i,t)\,dt$ into one summation:

$$\begin{aligned}
l(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) \quad \propto \quad & \left(\sum_i - \int_0^{\min(t_E^{(i)},T)} C(x_i,t)\,dt + \sum_{i;t_E^{(i)}\le T} \log C(x_i,t_E^{(i)})\right) + \\
& \sum_{i;t_I^{(i)}\le T} \log\left(f_E(t_I^{(i)} - t_E^{(i)};\alpha_E,\nu_E)\right) + \\
& \sum_{i;t_I^{(i)}> T} \log\left(1 - F_E(t_I^{(i)} - t_E^{(i)};\alpha_E,\nu_E)\right) + \\
& \sum_{i;t_R^{(i)}\le T} \log\left(f_I(t_R^{(i)} - t_I^{(i)};\alpha_I,\nu_I)\right) + \\
& \sum_{i;t_R^{(i)}> T} \log\left(1 - F_I(t_R^{(i)} - t_I^{(i)};\alpha_I,\nu_I)\right) \qquad (5.9.1)
\end{aligned}$$

Computing the first and second sums are the operations that will be performed on the GPU; the other four sums can involve complicated (for example, incomplete gamma) functions that are not available on the GPU. These sums will be computed on the CPU in parallel whilst the computation of the first two sums are calculated on the GPU and then added together.

The sum of the integrals need to be changed into a form that can be calculated on the GPU. The second sum is a map operation followed by a reduction operation (i.e. a summation), followed by a multiply and addition, so is already in the form needed for easy implementation on the GPU.

Let $i \in \{1,2,\ldots,N\}$:

$$\begin{aligned}
C(x_i,t) &= \alpha + \beta\sum_{j=1}^{N} \mathbf{1}_{t_E^{(i)}>t_I^{(j)}} \mathbf{1}_{y_j\in I(t)} K(\mathbf{x}_i,\mathbf{y}_j,\kappa) \qquad (5.9.2)\\
\Rightarrow \int_0^s C(x_i,t)\,dt &= \int_0^s \left(\alpha + \beta\sum_{j=1}^{N} \mathbf{1}_{t_E^{(i)}>t_I^{(j)}} \mathbf{1}_{y_j\in I(t)} K(\mathbf{x}_i,\mathbf{y}_j,\kappa)\right) dt\\
&= \alpha s + \beta\sum_{j=1}^{N} \mathbf{1}_{t_E^{(i)}>t_I^{(j)}} K(\mathbf{x}_i,\mathbf{y}_j,\kappa) \int_0^s \mathbf{1}_{y_j\in I(t)}\,dt\\
&= \alpha s + \\
&\quad \beta\sum_{j=1}^{N} \mathbf{1}_{t_E^{(i)}>t_I^{(j)}} K(\mathbf{x}_i,\mathbf{y}_j,\kappa) \max\left(0, \left(\min(s,t_R^{(j)}) - \min(s,t_I^{(j)})\right)\right)
\end{aligned}$$

Hence:

$$\int_0^{\min(t_E^{(i)},T)} C(x_i,t)\,dt = \alpha\min(t_E^{(i)},T) + \beta\sum_{j=1}^{N}\mathbf{1}_{t_E^{(i)}>t_I^{(j)}} K(\mathbf{x}_i,\mathbf{y}_j,\kappa)\max\left(0,\left(\min(t_E^{(i)},T,t_R^{(j)})-\min(T,t_I^{(j)})\right)\right) \quad (5.9.3)$$

These equations only involve functions that can be computed on the GPU. Substitute eq. 5.9.2 and eq. 5.9.3 into eq. 5.9.1:

$$\begin{aligned} l(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) \propto & \sum_i\Bigg[\alpha\min(t_E^{(i)},T) + \\ & \beta\sum_{j=1}^{N}\mathbf{1}_{t_E^{(i)}>t_I^{(j)}} K(\mathbf{x}_i,\mathbf{y}_j,\kappa)\max\left(0,\left(\min(t_E^{(i)},T,t_R^{(j)})-\min(T,t_I^{(j)})\right)\right)\Bigg] + \\ & \sum_{i;t_E^{(i)}\le T}\log\left(\alpha+\beta\sum_{j=1}^{N}\mathbf{1}_{t_E^{(i)}>t_I^{(j)}}\mathbf{1}_{y_j\in I(t)}K(\mathbf{x}_i,\mathbf{y}_j,\kappa)\right) + \\ & \sum_{i;t_I^{(i)}\le T}\log\left(f_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right) + \\ & \sum_{i;t_I^{(i)}>T}\log\left(1-F_E(t_I^{(i)}-t_E^{(i)};\alpha_E,\nu_E)\right) + \\ & \sum_{i;t_R^{(i)}\le T}\log\left(f_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right) + \\ & \sum_{i;t_R^{(i)}>T}\log\left(1-F_I(t_R^{(i)}-t_I^{(i)};\alpha_I,\nu_I)\right) \end{aligned}$$

We split the log likelihood into two parts for the purpose of computation:

$$l(\alpha,\beta,\kappa,\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x}) = l_1(\alpha,\beta,\kappa|\mathbf{x}) + l_2(\alpha_E,\nu_E,\alpha_I,\nu_I|\mathbf{x})$$

As stated earlier l_1 will be computed on the GPU whilst l_2 is being calculated on the CPU. Further simplification yields:

$$\begin{aligned}
l_1(\alpha, \beta, \kappa|\mathbf{x}) &= \sum_i \Bigg[\alpha \min(t_E^{(i)}, T) + \\
&\quad \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \max\left(0, \left(\min(t_E^{(i)}, T, t_R^{(j)}) - \min(T, t_I^{(j)})\right)\right) \Bigg] + \\
&\quad \sum_{i; t_E^{(i)} \le T} \log\left(\alpha + \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} \mathbf{1}_{y_j \in I(t)} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \right) \\
&= \sum_{i=1}^N \Bigg[\alpha \min(t_E^{(i)}, T) + \\
&\quad \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \max\left(0, \left(\min(t_E^{(i)}, T, t_R^{(j)}) - \min(T, t_I^{(j)})\right)\right) + \\
&\quad \mathbf{1}_{t_E^{(i)} \le T} \log\left(\alpha + \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} \mathbf{1}_{y_j \in I(t)} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \right) \Bigg] \\
&= \sum_{i=1}^N \Bigg[\alpha \min(t_E^{(i)}, T) + \\
&\quad \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \max\left(0, \left(\min(t_E^{(i)}, T, t_R^{(j)}) - \min(T, t_I^{(j)})\right)\right) + \\
&\quad \mathbf{1}_{t_E^{(i)} \le T} \log\left(\alpha + \beta \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} \mathbf{1}_{y_j \in I(t)} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \right) \Bigg]
\end{aligned}$$

Let:

$$\begin{aligned}
S_{1i} &= \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \cdot \max\left(0, \left(\min(t_E^{(i)}, T, t_R^{(j)}) - \min(T, t_I^{(j)})\right)\right) \\
S_{2i} &= \sum_{j=1}^N \mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa) \cdot \mathbf{1}_{y_j \in I(t)}
\end{aligned}$$

then

$$l_1(\alpha, \beta, \kappa|\mathbf{x}) = \sum_{i=1}^N \left[\alpha \min(t_E^{(i)}, T) + \beta S_{1i} + \mathbf{1}_{t_E^{(i)} \le T} \log\left(\alpha + \beta S_{2i}\right) \right]$$

This is simply a series of map and reduce operations, which is suited to computation on the GPU.

Observe in the above expression the outer sum can be evaluated in a massively parallel way, where each element of the sum is evaluated individually and combined into a final single value via a reduction.

Note from above certain features of the expression:

1. If κ remains constant between successive likelihood calculations (often the case in Metropolis-within-Gibbs), $K(\mathbf{x}, \mathbf{y}_j, \kappa)$ can be pre-calculated and re-used. Since $K(\mathbf{x}, \mathbf{y}_j, \kappa)$ often involves the numerical calculation of the `exp()` function, which is relatively lengthy, this transforms the task from a compute intensive task to a series of memory lookups and indicator functions.

2. There are a relatively few number of conditional statements, making it suitable for the GPU. Any conditional statements that remain are trivial and therefore quick to evaluate.

3. For each i in the outer sum above, there are several elements which are repeated. The spatial kernel $K(\mathbf{x}_i, \mathbf{y}_j, \kappa)$ and indicator $\mathbf{1}_{t_E^{(i)} > t_I^{(j)}}$ are sometimes used twice per evaluation.

4. Only the numerical result of $l_1(\alpha, \beta, \kappa|\mathbf{x})$ needs to be transferred back from the GPU minimising memory transfer back from the GPU.

5. All of the observed data can be kept on the GPU for future likelihood calculations. This minimises memory copy to the GPU.

6. Note that S_{1i} is an alternate form of

$$\sum_{i;t_E^{(i)} \leq T} -\int_0^{t_E^{(i)}} C(x_i, t)\, dt + \sum_{i;t_E^{(i)} > T} -\int_0^{T} C(x_i, t)\, dt$$

7. Note that S_{2i} is an alternate form of

$$\sum_{i;t_E^{(i)} \leq T} \log C(x_i, t_E^{(i)})$$

Note that from above, it is possible to also parallelise S_{1i} and S_{2i} as well as the outer sum, evaluating each element on different threads within a block and performing a reduction. In fact it is possible to evaluate both a single element of the sum S_{1i} and a single element of the sum S_{2i} on a single thread, by first evaluating $\mathbf{1}_{t_E^{(i)} > t_I^{(j)}} K(\mathbf{x}_i, \mathbf{y}_j, \kappa)$ then multiplying by $\max\left(0, \left(\min(t_E^{(i)}, T, t_R^{(j)}) - \min(T, t_I^{(j)})\right)\right)$ and $\mathbf{1}_{y_j \in I(t)}$ to get an element of the sum of S_{1i} and S_{2i} respectively. This approach is taken in the code for this thesis.

5.9.1 Layout of the thread grid and blocks

The thread blocks will need to therefore be 2-dimensional to implement the two layers of summation above. The thread grid will be one block tall, and several blocks wide (depending on the size of the data). Each thread in each thread block will compute an element of the sums S_{1i} and S_{2i}, the results in each column will be reduced via a reduction algorithm to obtain an element of the outer sum. After each column in each block has evaluated an element of the outer sum, the outer sum will be evaluated by performing a reduction.

The grid consists of m_b thread blocks, which have the 2D arrangement:

$$block_1 \quad block_2 \quad block_3 \quad \ldots \quad block_{m_b}$$

Each block consists of $m_{t_x} \times m_{t_y}$ threads which are arranged in the following 2D configuration, where m_{t_y} is always a power of two (the cascading reduction described earlier will be used to reduce the inner sum):

$$\begin{matrix} thread_{11} & thread_{12} & thread_{13} & \cdots & thread_{1m_{t_x}} \\ thread_{21} & thread_{22} & thread_{23} & \cdots & thread_{2m_{t_x}} \\ thread_{31} & thread_{32} & thread_{33} & \cdots & thread_{3m_{t_x}} \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ thread_{m_{t_y}1} & thread_{m_{t_y}2} & thread_{m_{t_y}3} & \cdots & thread_{m_{t_y}m_{t_x}} \end{matrix}$$

Note that the same time, as the GPU calculations are being completed, the CPU calculates $l_2(\alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x})$. This calculation is calculated in parallel on the CPU (a simple map reduce operation). This result is added to the result from the GPU to obtain the full likelihood $l(\alpha, \beta, \kappa, \alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x}) = l_1(\alpha, \beta, \kappa|\mathbf{x}) + l_2(\alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x})$. The calculation on the CPU is because the incomplete gamma function is not available on the GPU (at the time of implementation).

5.10 The CUDA kernels

Included here is the CUDA code for the GPU kernels. The CPU code is not included here, since it comprises several hundred lines of code, mostly comprised of boilerplate code (verbose code that performs basic but essential tasks, for example memory transfer to and from the GPU, interface with the numerical optimisation library, interface between the C++ and CUDA code, refreshing the data on the GPU, conversion between the parametrisation used in different libraries etc.). Readers are not expected to understand fully the code presented in this section, as it would require an understanding of the thousands of lines of code not related to the material in this chapter. An explanation of what this code does is detailed in section 5.11. An explanation will be given in the following sections, with references to the relevant line numbers.

5.10.1 Precomputation and storage of the transmission kernel

```
1 __global__ void
2 SLogLKern2cached_general_updatekern(const double alpha, const double
      beta, const double kappa, const double Ealpha, const double Ebeta,
      const double Ialpha, const double Ibeta, const double* __restrict__
       d_Etime, const double* __restrict__ d_Itime, const double*
      __restrict__ d_Rtime, double* __restrict__ d_tempout, const double
      time_T, const int N, double* d_currentkappa, double* __restrict__
      d_cachedtrkern, int* d_kerneltype, int kerneltype, double*
      __restrict__ d_cacheddist, double* d_loaddist, double* __restrict__
       d_dirmatx, double* __restrict__ d_dirmaty, const double*
      __restrict__ wx, const double* __restrict__ wy, const double*
      __restrict__ tw, const int windtype, int* __restrict__ d_windtype)
3 {
4     double trkern;
5     double dr_cachedist;
6     double dr_xcoordj;
```

```
    double dr_ycoordj;
    double dr_dx;
    double dr_dy;


    if ((__ldg(d_currentkappa) != kappa) || (__ldg(d_kerneltype) !=
        kerneltype) || (dr_loaddist != 0.0) || (__ldg(d_windtype) !=
        windtype))
    {

        for (MYINT j = blockIdx.x * blockDim.x + threadIdx.x;
            j < POPSIZE;
            j += blockDim.x * gridDim.x)
        {
            if (dr_loaddist != 0.0)
            {
                dr_xcoordj = d_xcoord[j];
                dr_ycoordj = d_ycoord[j];
            }
            for (MYINT i = blockIdx.y * blockDim.y + threadIdx.y;
                i < j;
                i += blockDim.y * gridDim.y)
            {

                if (dr_loaddist != 0.0)
                {
                    double t1 = d_xcoord[i] - dr_xcoordj;
                    double t2 = d_ycoord[i] - dr_ycoordj;
                    dr_cachedist = sqrtf(t1*t1 + t2*t2);

                    d_cacheddist[POPSIZE*j + i] = dr_cachedist;
                }
                else
                {
                    dr_cachedist = __ldg(&d_cacheddist[POPSIZE*j + i]);
                }

                if (kerneltype == 0)
                {

                    trkern = expf(-kappa*dr_cachedist);

                }
                else if (kerneltype == 1)
                {
                    trkern = 1.0f / (1.0f + powf((float)dr_cachedist, (float)kappa));

                }
                else if (kerneltype == 2)
                {
                    trkern = powf(1.0f + (float)(dr_cachedist / kappa), -1.0f);

                }
                else if (kerneltype == 3)
                {
                    trkern = 1.0 / (1.0 + (dr_cachedist*dr_cachedist / kappa));
                }
                else if (kerneltype == 4)
                {
                    trkern = 1.0f / (1.0f + (float)(dr_cachedist * kappa));
                }
                else if (kerneltype == 5)
                {
                    trkern = expf(-kappa*dr_cachedist*dr_cachedist);
                }
                else
                {
                    printf("ERROR kernel");
                }
                d_cachedtrkern[POPSIZE*j + i] = trkern;
```

```

            }
        }
    }
}
```

5.10.2 Calculation of $l_1(\alpha, \beta, \kappa|\mathbf{x})$ - Parts of the Log-Likelihood that Can Be Calculated on the GPU

```
__global__ void SLogLKern2cached_aftergpucourse5_2cachedist2(const
    double alpha, const double beta, const double kappa, const double
    Ealpha, const double Ebeta, const double Ialpha, const double Ibeta
    , const double* __restrict__ d_Etime, const double* __restrict__
    d_Itime, const double* __restrict__ d_Rtime, double* __restrict__
    d_tempout, const double time_T, const int N, double* d_currentkappa
    , double* __restrict__ d_cachedtrkern, int* d_kerneltype, int
    kerneltype, double* __restrict__ d_cacheddist, double* d_loaddist,
    const int calce)
{
    double trkern;
    double dr_Etimej;

    if (blockIdx.y == 0 && blockIdx.z == 0)
    {
        __shared__ double sharedtemp[BLOCKV][BLOCKH];
        __shared__ double sharedtemp2[BLOCKV][BLOCKH];

        for (MYINT j = blockIdx.x * blockDim.x + threadIdx.x;
            j < POPSIZE;
            j += blockDim.x * gridDim.x)
        {

            double temp = 0.0;
            double temp2 = 0.0;

            dr_Etimej = __ldg(&d_Etime[j]);

            for (MYINT i = threadIdx.y;
                i < j;
                i += blockDim.y)
            {
                if (dr_Etimej > d_Itime[i])
                {
                    trkern = __ldg(&d_cachedtrkern[POPSIZE*j + i]);

                    if (dr_Etimej < d_Rtime[i] && dr_Etimej <= time_T
                        && d_Itime[i] <= time_T)
                    {
                        temp2 += trkern;
                    }
                    temp += fmax(0.0, (fmin(time_T, fmin(dr_Etimej,
                        d_Rtime[i])) - fmin(time_T, d_Itime[i])))*
                        trkern;
                }
            }

            sharedtemp2[threadIdx.y][threadIdx.x] = temp2;
            sharedtemp[threadIdx.y][threadIdx.x] = temp;


            __syncthreads();



            for (MYINT i = BLOCKV / 2; i > 0; i >>= 1)
            {
                if (threadIdx.y < i){
                    sharedtemp[threadIdx.y][threadIdx.x] += sharedtemp[
                        threadIdx.y + i][threadIdx.x];
                    if (dr_Etimej <= time_T){
                        sharedtemp2[threadIdx.y][threadIdx.x] +=
                            sharedtemp2[threadIdx.y + i][threadIdx.x];
                    }
                }
                __syncthreads();
            }


            if (threadIdx.y == 0)
            {
                if (dr_Etimej <= time_T)
                {
                    d_tempout[j] = log(alpha + beta*sharedtemp2[0][threadIdx.x]) - (alpha*fmin(dr_Etimej, time_T) + beta*sharedtemp[0][threadIdx.x]);
                }
                else
                {
                    d_tempout[j] = -(alpha*fmin(dr_Etimej, time_T) + beta*sharedtemp[0][threadIdx.x]);
                }
            }


        }


    }
}
```

5.11 Implementing the algorithm

We implement the RJMCMC to make use of the heterogeneous structure of the computer. Random number generation and the choice to accept/reject proposals are done on the CPU. In addition, there are some parts of the likelihood which cannot be computed on the GPU, specifically the incomplete gamma function.

The likelihood is implemented as thus:

1. The data are copied to the GPU and any precomputation is performed

(Kernel "`SLogLKern2cached_general_updatekern`"):

   (a) When the likelihood subroutine is called, the program on the host checks whether the data are up to date, otherwise the data are uploaded to the GPU (CPU code, GPU code has an "if" statement to double check at line 12).

   (b) The distance matrix for all the hosts in the epidemic is precomputed and stored. This is only updated if the data-set has been changed (lines 29-36).

   (c) The transmission kernel $K(\mathbf{x}_i, \mathbf{y}_j, \kappa)$ is pre-computed for all pairs of hosts i and j in the epidemic, using the distance matrix in the last step. The distances are read in from Global memory through the texture pipeline to speed up the loads. Adjacent threads access adjacent memory locations, making use of "burst utilisation", described in section 5.6.1 on page 139). The results are stored in an array (line 74). It is not computed if up to date. This is a parallel map (described in section 26 on page 137) .

2. The task of calculating S_{1ij} and S_{2ij} is performed by column j, with the task divided up between the threads in the column (Kernel "`SLogLKern2cached_aftergpucourse5_2cachedist2`"):

   (a) Each thread column in each block calculates several of the S_{1ij}'s and the S_{2ij}'s, each thread in a column performing its own privatised calculation (this is privatisation, as in section 5.6.4 on page 142, with granularity coarsening, one of the general optimisation techniques for parallel programming). These are summed in each thread to form a temporary sum.

   (b) There is a thread barrier where processors that have finished their calculations so far have to wait upon processors which are still calculating.

   (c) The privatised results are merged into block-wide results: When all processors have reached the barrier, a simple cascading reduce (described in section 5.5 on page 136) is performed on the temporary sums stored by each thread column (this is why the number of each threads in a column in each block are always a power of two). This yields S_{1i} and S_{2i}.

3. Each element of the outer sum is computed: that is, $E_i = \alpha \min(t_E^{(i)}, T) + \beta S_{1i} + \mathbf{1}_{t_E^{(i)} \leq T} \log\left(\alpha + \beta S_{2i}\right)$ is calculated. This is a parallel map (described in section 26 on page 137) . These are written to global DRAM. Adjacent threads write to adjacent memory locations known as *memory coalescing*, described in section 5.6.1 on page 139. The threads are synchronised at a barrier. This yields an array containing the E_i's which is situated on the GPU (this is privatisation, as in section 5.6.4 on page 142).

4. As there is no way to form a barrier to synchronise all thread blocks on the GPU without the intervention of the CPU, control is returned to the CPU, which then waits for all computation to complete on the GPU and launches a kernel on the GPU which computes $l_1(\alpha, \beta, \kappa|\mathbf{x}) = \sum_{i=1}^{N} E_i$. This is a global reduction, implemented in the library CUDA CUB. During compilation, the library determines what GPU is fitted in the computer and selects the optimal reduction algorithm. This may comprise one or many kernels, depending on what is optimal for the model GPU fitted in the computer. This result is then transferred from the memory of the GPU, back to the CPU.

5. At the same time, the CPU has been calculating $l_2(\alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x})$, by the standard method. This calculation is simply a parallel map followed by a reduction, and is parallelised using OpenMP. There is a barrier where the computer has to wait for both the GPU and CPU calculations to finish.

6. This result is added to the result from the GPU to obtain the full likelihood $l(\alpha, \beta, \kappa, \alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x}) = l_1(\alpha, \beta, \kappa|\mathbf{x}) + l_2(\alpha_E, \nu_E, \alpha_I, \nu_I|\mathbf{x})$.

5.12 Optimisation of the algorithm

Readers should note that the word "optimisation" is used in the programming sense: to improve performance. This does not mean that the algorithm detailed here is the ultimate algorithm for its purpose, and that no improvement is possible. However, the algorithm detailed here gives high performance, whilst leaving the code relatively straightforward to understand, and adequately flexible the work that was done for this thesis.

Several optimisation patterns detailed in section 5.6 have been applied to the code. *Privatisation* was applied, so that each thread will calculate its own partial sum, with the final partial sums aggregated into one global result. The calculation of the log-likelihood is a *gather* operation. Temporary data structures were held in fast temporary memory, with read-only loads from global memory going through the texture buffer. Using the CUDA CUB library allowed the automatic selection of *reduction* algorithm in the device-wide *reduction* dependent on what hardware was available and the size of the data. The data were sorted to avoid warp divergence.

5.13 Implementing the ILR residuals test on the GPU

For the initial test runs of the ILR, the algorithm was performed on the CPU. However, for the FMD 2001 data-set, the data-set was so large (almost 200,000 hosts) that it took over 3 hours for 1 iteration of the algorithm. This algorithm was ported to the GPU to speed up the calculation.

There are two major challenges in implementing the algorithm with GPGPU programming. The first of these challenges is the size of the data: usually it helps to implement a CPU version of the algorithm first, then move it step-by-step onto the GPU. Since the CPU version of the algorithm would crash because of the size of the data, it was difficult to debug the GPU version. Intermediate results from the CPU algorithm (sometimes with subsets of the data), were taken and compared to the GPU algorithm for the purposes of debugging. In addition, since the data was so large, it required a new data layout to conserve space. With this data layout, it was not straightforward to calculate the amount of memory needed on the GPU, which needs to be specified when allocating memory on the GPU. The new data layout also need implementation to be coded by hand, as there was no library which would give the functionality needed.

The second challenge was that the operations required to calculate the ILR were not particularly suited to the GPU. In particular, there is a step in the algorithm which requires sorting. However, an equivalent method was found that did not require sorting, circumventing this challenge.

5.13.1 Sparse Matrices

The transmission kernel was assumed to be 0 outside a radius of 25km. This made it possible to pre-compute and store the transmission matrix in CSR format, a format for storing sparse matrices (matrices where the majority of entries are 0).

Definition 27 (CSR matrix format)**.** This stores a $m \times n$ matrix M as three arrays $(\mathtt{A}, \mathtt{IA}, \mathtt{JA})$. `A` stores all the non-zero entries of M. Let m_{ij} be a non-zero entry of M. The zero entries of m_{ij} are not stored. Suppose a program requires data m_{ij} from the matrix M which is stored as $(\mathtt{A}, \mathtt{IA}, \mathtt{JA})$. The ith element of `IA` holds the index of the first element of `A` that is in row i. For example, if the value of $m_{ij} = 1$ were stored at $\mathtt{A}[\mathtt{k}]$, then $\mathtt{JA}[\mathtt{k}]$ would be j, and $\mathtt{A}[\mathtt{k}] = 1$.

Thus, to find element m_{ij}:

1. Determine $\mathtt{IA}[i]$ and $\mathtt{IA}[i+1]$.
2. Search $\mathtt{JA}[\mathtt{IA}[i]]$ to $\mathtt{JA}[\mathtt{IA}[i+1]]$ using a search algorithm (for example bisection search). Suppose this yields the result p (when $m_{ij} \neq 0$). If the search does not find any result, $m_{ij} = 0$.
3. Retrieve element $\mathtt{A}[\mathtt{IA}[i] + p]$.

Readers will observe that it is an inefficient process to retrieve an element from a matrix stored in this format. This is why the algorithms implemented in this chapter avoid looking up values at a given column index, and instead, from a given entry $\texttt{A}[\texttt{IA}[i] + p]$ at a given row index i, determine the column index by retrieving $\texttt{JA}[\texttt{IA}[i] + p]$.

5.13.1.1 Pre-computing the transmission kernel matrix

Memory management on the GPU is allocated and deallocated by the user. Memory allocations cannot be altered in size once they are re-allocated. Thus, the required size of A is calculated on the CPU by determining the pairs of plots of land less than 25 km of each other. Since this does not change between iterations, the amount of space is determined from this calculation and allocated. The transmission kernel is pre-calculated by:

1. For $i = 1 \ldots n$:
    (a) For $p = 1 \ldots (\texttt{IA}[i + 1] - \texttt{IA}[i])$:
        i. Set $j = \texttt{JA}[\texttt{IA}[i] + p]$.
        ii. Store $K(\mathbf{x}_i, \mathbf{y}_j, \kappa)$ at $\texttt{A}[\texttt{IA}[i] + p]$.

5.13.2 Inferring the SI link

All the possible SI links for a plot of land at $\mathbf{x}_i$ are stored in $\texttt{A}[\texttt{IA}[i] + p]$ where $p = 1 \ldots (\texttt{IA}[i+1] - \texttt{IA}[i])$. From this sub-array, we can determine the index of the potential infectors $j = \texttt{JA}[\texttt{IA}[i] + p]$. From this we can filter these links such that each of the $\mathbf{y}_j$ remaining are infective before $\mathbf{x}_i$ and removed only after $\mathbf{x}_i$ has been infected. This yields an array KA of all the potential infectors of $\mathbf{x}_i$. Note that a link is added for a notional infector corresponding to primary infection. A random deviate from the uniform distribution $r \sim \text{Unif}(0, \sum_{i=1}^{\texttt{Size(KA)}} \texttt{KA}[i])$ is generated, and then used to select a link from this sub-array, such that the link selected is the greatest link k less than $\sum_{i=1}^{k} \texttt{KA}[i]$. The corresponding link k selected is the inferred infection link. Let the infector in this susceptible-infective pair be denoted j'.

5.13.3 Determining rank

For each pair of hosts (l, j), recall that in an earlier step $K(\mathbf{x}_l, \mathbf{y}_j, \kappa)$ has been pre-computed. In this step, to determine all possible SI pairs at the time of infection, the aforementioned array is filtered such that only the possible SI links at the time of infection of i remain (that is for each j is the filtered array, j is infective and unremoved before the infection of i). This yields the array `KA2`.

Now in the definition of the algorithm specified earlier in this thesis (see Section 3.3.2 on page 75), there is a sorting stage in which these links are sorted and the rank of the imputed SI link is determined in this array. Note that on the GPU sorting is a relatively difficult operation to implement, but sorting can be avoided by implementing an equivalent operation. First of all note, that they are three corresponding classes of link: the links in `KA2` that are smaller in value than the imputed SI link, the links in `KA2` which are larger in value than the imputed SI link, and the links in `KA2` which are the same in value as imputed SI link. From the sub-array obtained earlier, the possible SI links can be categorised in parallel into these three groups. Suppose that values of the links which fall into each category are summed and stored in $\mathtt{m}[0], \mathtt{m}[1], \mathtt{m}[2]$ respectively. To generate the inferred infection link residual, a uniform random deviate is generated

$$r_2 \sim \text{Unif}(0, 1)$$

From this the imputed SI link can be determined by computing:

$$\frac{\mathtt{m}[0] + \mathtt{m}[1] \cdot r_2}{\mathtt{m}[0] + \mathtt{m}[1] + \mathtt{m}[2]}$$

thus avoiding a sorting step. This also lowers the complexity of the algorithm, which can escalate when the data-set is large, as in the case of foot-and-mouth disease. The infection link is then stored in an array on the host CPU. Once all the infection link residuals have been calculated the p-value can be determined using the usual calculation for the Anderson-Darling test p-value [116].

Chapter 6

Directional Infection Link Residuals Tests

One of the strengths of the functional-model approach for creating test statistics is that as long as the sampling distribution for the whole model is preserved the residuals can be specified to focus on various aspects of mis-specification of the model. In many cases, the tail properties of the spatial kernel has been of vital importance because of its relation to culling radius. There have been several instances where mathematical modelling has influenced culling radius (for example, [100, 48, 49]) or whether it is possible (under budget constraints) that a ring culling strategy can be put in place at all (for example, [40]). Another underlying assumption of many models is the isotropic assumption, where the kernel is not dependent on the bearing from one host to another, but only the length of the vector joining them. Such an assumption may be unsafe to make under certain circumstances as there may be reason for the disease not to spread evenly in all directions. Wind direction, local geography, and other phenomena may cause infection spread in one direction more than another (for example, [51, 152]). Such model mis-specification may cause the tail length of the kernel to be misestimated. Therefore, it is desirable to have a test for all the various types of mis-specification that would lead to the choice of wrong control strategy.

As well as affecting control strategy, selection of spatial kernel can affect surveillance strategy [4, 3], in identifying which hosts to survey in order to best detect occurrence of the disease.

In this chapter, tests for anisotropy are formulated and analysed using simulated data to determine the relative power of the various test statistics which have been developed for anisotropy. Here, we propose two variants of the infection link residual test that are designed to detect model mis-specification arriving from anisotropy of a spatial kernel. Specifically, we consider test that can be applied in the absence of an alternate (anisotropic) model.

6.1 The Modelling of Anisotropy in Compartmental Spatio-temporal Epidemic Models

The most straightforward way of representing anisotropy in the spatial transmission of the epidemic is through incorporation of the anisotropy in the spatial transmission kernel, as this represents the infectious challenge between an infected host and a susceptible host. In this case, the spatial transmission kernel is a function of angle of the infectious contact as well as the distance of the infectious contact. The angle may be measured from a fixed angle or the prevailing wind direction at the time of the infectious contact. In some cases, how the force of infection varies proportionately by distance may be separate from how force of infection varies by angle, giving a kernel function that is separable into independent distance varying and angle varying components.

6.1.1 Formal Description of the Model

More formally, as in Section 2.1 on page 11, let $S(t), E(t), I(t), R(t)$ be the set of hosts in the relevant state at time t. Hosts transition from $S \rightarrow E$, $E \rightarrow I$ and $I \rightarrow R$. The hosts (which are indexed by $1, 2, \ldots, N$) are distributed over a 2-dimensional region at known coordinates $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where the population is of a known size N. Hosts in state S at time t, $S(t)$ experience infectious challenge from two sources: primary infectious challenge, from sources/sites external to the system under study, and secondary infectious challenge, from infectious individuals within the system. Let α and β be the primary and secondary infection rates. Then the probability of exposure (for an arbitrary host $x \in S(t)$) can be modelled by the following equation:

$$\begin{aligned}\Pr(j \text{ exposed during } [t, t+dt]) &= C(\boldsymbol{x}_j, t)\, dt + o(dt) \\ &= \left(\alpha + \beta \sum_{i \in I(t)} K(\boldsymbol{x}_j, \boldsymbol{x}_i, \kappa)\right) dt + o(dt)\end{aligned} \tag{6.1.1}$$

The function $K(\boldsymbol{x}, \boldsymbol{y}, \kappa)$ in equation 6.1.1 is known as the transmission kernel, which models the effect of distance and angle on the infectious challenge from each infectious host to **x**. In previous chapters, models have featured such transmission kernels as the *exponential kernel* $K(\boldsymbol{x}, \boldsymbol{y}, \kappa) = \exp\{-\kappa|\boldsymbol{x} - \boldsymbol{y}|\}$ and the *power-law kernel* $K(\boldsymbol{x}, \boldsymbol{y}, \kappa) = (1 + |\boldsymbol{x} - \boldsymbol{y}|^{\kappa})^{-1}$ (used for example in [123, 134, 36, 38]) where $|\cdot|$ is the Euclidean distance. These kernels, being isotropic, are functions of the distance metric between the coordinates of the S-I pair, $\boldsymbol{x}$ and $\boldsymbol{y}$, $d(\boldsymbol{x}, \boldsymbol{y}) = |\cdot|$. This principle can be extended by allowing the transmission kernel to be related to the angle between the vector between the S-I pair, $\boldsymbol{y} - \boldsymbol{x}$, and some vector $\boldsymbol{v}$: $\varphi(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{v})$. Analogous to the expression of an isotropic kernel being expressed in some papers as a function of distance, $K(d, \kappa)$, an anisotropic kernel can be expressed as a function of distance and angle $K(d, \varphi, \kappa)$.

6.1.2 Extension of the Infection Link residuals Test to Test for Discrepancy from the Assumption of Isotropy

As shown in [111], the formulation of the infection link residuals $\tilde{r}_2$ shares the same marginal model $\pi(\theta|y)$ regardless of whatever ordering is used for the infection links. Thus, the infection link residuals test can be made to detect anisotropy if an ordering of the infection links can be found that is sensitive to mis-specification of the isotropy assumption.

If the infection links were to be ordered by ϕ or $\cos(\phi)$, the angle between the infection links and the vector $(1, 1)$, if the assumption of isotropy were correct, the imputed directional infection link residuals would be distributed $\text{Unif}(0, 1)$. If the assumption of isotropy were incorrect, then this would produce a clustering of imputed infection links at certain angles, and thus produce clustering of imputed directional infection link residuals at certain sections of the unit interval, and thus reveal any discrepancy. Thus, the infection link residual test can extended to the directional infection link residual test, by the following definition:

Definition 28 (Directional Infection link residuals)**.** Suppose that host j is infected by infectious host i and that this is the kth exposure event. The infection link residual for this infection time is defined by the following:

Let the kth exposure event be between hosts i and j with probability $p_{ij} \propto \beta K(\mathbf{x}_i, \mathbf{x}_j, \kappa)$. Primary infection is treated as infection from a notional infector with force of infection α.

For all $m \in S(t_k)$ and $n \in I(t_k)$, let $p_{mn} \propto \beta K(\mathbf{x}_m, \mathbf{x}_n, \kappa)$.

Order the p_{mn} such that $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the cosine of the angle between the infection link and the vector $(1, 1)$.

Let s' be such that $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf \left\{ s | \tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)} \right\} = s'$$

We also used an alternate version where $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the **angle** between the infection link and the vector $(1, 1)$.

The cosine of the angle can be interpreted as the projection of inoculum blown by wind in the direction from the infective to the susceptible, if that fixed angle used is that of wind direction. Rather than a fixed direction vector, the angle can be taken between the wind direction at the time of the infection and the infection link, if wind direction data is available. However, wind direction is a complicated phenomenon, altered by many variables such as height above ground level, and other local geography, including buildings that may stand in the way of wind.

These variables may often not be recorded. In addition, the wind direction is not available at all points, and is instead interpolated from data from nearby weather stations using a meteorological model, which may introduce errors in terms of the wind direction. For this reason, it may be advisable to use a fixed direction vector to measure the angle of infection links against. The prevailing wind direction in the UK is generally accepted as being South-West, so the vector $(1, 1)$ is a reasonable choice of reference direction. The use of ordering the infection links by the angle between the infection link and the direction vector instead of using the cosine of the angle is motivated by the fact that the cosine function is an even function, and hence by transforming the angle by the cosine function may be losing some information about the angle. Primary infections are treated as infectious challenge incoming from a notional infective, such that the angle between the infection link and the vector $(1, 1)$ is distributed Unif$(0, 2\pi)$.

6.2 Methodology

Runs were performed to determine the ability of the Directional Infection Link Residuals test to detect anisotropy versus Infection Link Residuals test. To calculate a p-value the Anderson-Darling test was used on the residuals. We also used the Kuiper test (to be described in the following section), as this test is rotation invariant, as this may be a more powerful test if the residuals are sorted by angle.

Extensive code modification was necessary to implement the Directional Infection Link residuals test.

The data were simulated from the Gillespie-like algorithm in Section 2.2 on page 13 with the following known parameters:

$$\begin{aligned}
\alpha &= 0.001000\\
\beta &= 3.000000\\
\kappa &= 0.030000\\
\mu_E &= 5.000000\\
\sigma_E^2 &= 2.500000\\
\mu_I &= 1.772450\\
\sigma_I^2 &= 0.858407\\
N &= 1000
\end{aligned}$$

The distributions of the waiting times for states E and I are gamma distributions, parametrised by their means μ_E, μ_I and variances σ_E^2, σ_I^2. The exposure times were unobserved, but the infection and removal times were observed. The host locations were uniformly distributed over a square region. The epidemic was observed up to $T = 50$ time units, that is, $S \longrightarrow E$ transitions are not observed, $E \longrightarrow I$ and $I \longrightarrow R$ transitions inside the interval $[0, T]$ are observed. The data in each simulated dataset was generated using an anisotropic exponential kernel, an isotropic exponential kernel was fitted to the data in order to assess the test's ability to detect model mis-specification of the isotropy assumption.

An anisotropic exponential kernel was used to generate the data. We base our choice of anisotropic kernel upon the paper [152]. In this paper, the researchers estimate of the parameters for the kernel:

$$K(d, \varphi) = \frac{f(\varphi)}{(g(\varphi))^2} \exp\left\{\frac{-d}{g(\varphi)}\right\}$$

where (d, φ) denotes the polar coordinates of a susceptible relative to an infective:

$$\begin{aligned}
f(\varphi) &\propto \exp\left\{\delta \cos(\varphi - \mu)\right\}\\
g(\varphi) &\propto \exp\left\{\kappa \cos(\varphi - v)\right\}
\end{aligned}$$

The estimates obtained were:

$$\begin{aligned}\hat{\delta} &= 2.28\\ \hat{\kappa} &= 1.18\\ \hat{\mu} &= 3.974\\ \hat{\nu} &= 3.71\end{aligned}$$

for Conidia spore dispersal for the fungus *Mycoshaerella fijiensis* (where angles have been converted to radians). Conidia spores are spore which are asexually produced spores produced from lesions in the diseased plants (there are also sexually produced spores produced only at the latest phase in the disease). The researchers used lines of trap banana plants situated every 2.5m up to 25m in each of the eight directions (North, North-West, West etc.) from a single centrally situated diseased plant, to obtain data to fit the dispersal kernel via maximum likelihood estimation.

In the kernel above, there is normalisation term which ensures that it integrates to 1. The kernels used in this thesis are not normalized. In order to see the relationship between the isotropic exponential kernel used in earlier the chapters of this thesis, the equation can be rearranged to make the comparison easier:

$$\begin{aligned}K(d,\varphi) &\propto \frac{\mathrm{e}^{\hat{\delta}\,\cos(\varphi-\hat{\mu})}}{(\mathrm{e}^{\hat{\kappa}\,\cos(\varphi-\hat{\nu})})^2}\exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\,\cos(\varphi-\hat{\nu})}}\kappa\right\}\\ &= \exp\left\{\hat{\delta}\,\cos(\varphi-\hat{\mu})-2\hat{\kappa}\,\cos(\varphi-\hat{\nu})\right\} && (6.2.1)\\ &\quad\times\exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\,\cos(\varphi-\hat{\nu})}}\kappa\right\} && (6.2.2)\end{aligned}$$

Since:

$$\begin{aligned}
\hat{\delta}\ \cos(\varphi - \hat{\mu}) - 2\hat{\kappa}\ \cos(\varphi - \hat{\nu}) &= \Re\mathfrak{e}(\hat{\delta}\ \exp i(\varphi - \hat{\mu}) \\
&\quad -2\hat{\kappa}\ \exp i(\varphi - \hat{\nu})) \\
&= \Re\mathfrak{e}(\exp(i\varphi)(\hat{\delta}\ \exp(-\hat{\mu} i) \\
&\quad -2\hat{\kappa}\ \exp(-\hat{\nu} i))) \\
&= \Re\mathfrak{e}(\exp(i\varphi)(\hat{\delta}\,(\cos(-\hat{\mu}) \\
&\quad +i\sin(-\hat{\mu})) \\
&\quad -2\hat{\kappa}\,(\cos(-\hat{\nu}) + i\sin(-\hat{\nu})))) \\
&= \Re\mathfrak{e}(\exp(i\varphi)(\left[\hat{\delta}\cos(-\hat{\mu}) - 2\hat{\kappa}\cos(-\hat{\nu})\right] \\
&\quad +i\left[\hat{\delta}\sin(-\hat{\mu}) - 2\hat{\kappa}\sin(-\hat{\nu})\right]) \\
&= \Re\mathfrak{e}(\exp(i\varphi)\times \\
&\quad (A\exp(-Bi))) \\
&= \Re\mathfrak{e}(A\exp(i(\varphi - B))) \\
&= A\ \cos\left(\varphi - B\right)
\end{aligned}$$

where

$$\begin{aligned}
A &= \sqrt{\left[\hat{\delta}\cos(-\hat{\mu}) - 2\hat{\kappa}\cos(-\hat{\nu})\right]^2 + \left[\hat{\delta}\sin(-\hat{\mu}) - 2\hat{\kappa}\sin(-\hat{\nu})\right]^2} \\
&\approx 0.6158304744 \\
B &= \arctan\left(\frac{\hat{\delta}\sin(-\hat{\mu}) - 2\hat{\kappa}\sin(-\hat{\nu})}{\hat{\delta}\cos(-\hat{\mu}) - 2\hat{\kappa}\cos(-\hat{\nu})}\right) \\
&\approx 5.541930711
\end{aligned}$$

Hence:

$$
\begin{aligned}
K(d,\varphi) &\propto \exp\left(\hat{\delta}\ \cos(\varphi-\hat{\mu}) - 2\hat{\kappa}\ \cos(\varphi-\hat{\nu})\right) \\
&\quad \times \exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\ \cos(\varphi-\hat{\nu})}}\kappa\right\} \\
&= \exp\left(A\cos\left(\varphi-B\right)\right) \\
&\quad \times \exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\ \cos(\varphi-\hat{\nu})}}\kappa\right\}
\end{aligned}
$$

Observe that there are two cosine terms in the kernel given here: the first term controls how the maximum infectious force varies by angle, the second term controls the tail length of the kernel in any given direction. These two components have a phase difference between them of approximately $\frac{\pi}{2}$.

To make the form of the anisotropic kernel easier to compare with the forms of the isotropic kernels used in previous chapters, anisotropic kernels used in this chapter will be expressed in this form. We choose as the first anisotropic kernel the kernel above, without the phase shifts for simplicity:

$$
K(d,\phi) \quad = \quad \exp\left\{A\ \cos\left(\phi\right)\right\}\exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\ \cos(\phi)}}\kappa\right\} \tag{6.2.3}
$$

where ϕ is the angle between the vector between the susceptible and infective and the vector $(1,1)$.

This is shown in fig. 6.2.1. Compare this with the isotropic exponential kernel in fig. 6.2.2.

An additional version of the anisotropic kernel was used which models a $\frac{\pi}{2}$ phase shift between the two cosine terms as referred to above.

$$
K(d,\phi) = \exp\left\{0.61583\ \cos\left(\phi\right)\right\}\exp\left\{\frac{-d}{\mathrm{e}^{\hat{\kappa}\ \sin(\phi)}}\kappa\right\} \tag{6.2.4}
$$

This is plotted in fig. 6.2.3.

Simulated data produced using these two kernels are shown in fig. 6.2.4 and fig. 6.2.5. Note the subtle difference between the epidemics produced by the

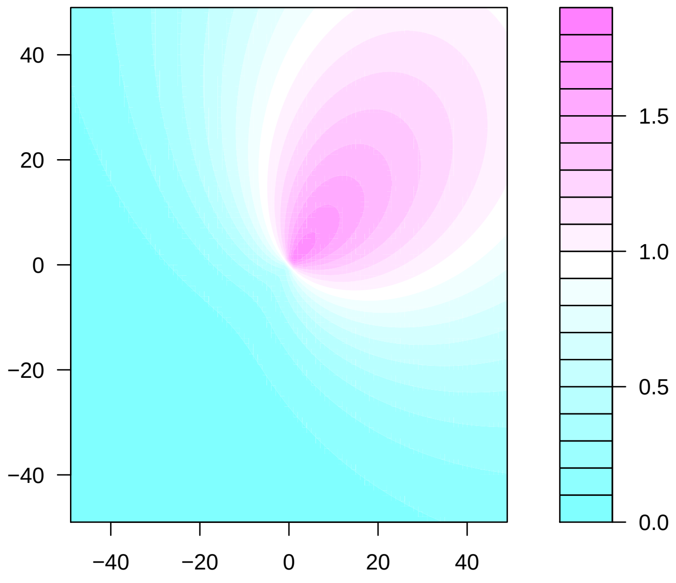

Figure 6.2.1: Heat-Plot of the anisotropic kernel given by Equation 6.2.3

two different kernels, the second kernel produces anisotropy which seems more difficult to detect with the naked eye.

6.2.1 Kuiper Test

The Kuiper test [106] tests the following hypotheses:

$$\begin{aligned} H_0 &: X_1, X_2, \ldots, X_n \text{ are i.i.d with distribution function } F(x) \\ H_A &: X_1, X_2, \ldots, X_n \text{ are not i.i.d with distribution function } F(x) \end{aligned}$$

The test statistic used is:

$$V = \max_i \left[\frac{i}{n} - F(x_i) \right] + \max_i \left[F(x_i) - \frac{i-1}{n} \right]$$

where the sample $x_1, x_2, \ldots, x_n$ is sorted such that $x_1 \leq x_2 \leq \ldots \leq x_n$. Note the similarities between the Kuiper test statistic and Kolmogorov-Smirnov test.

The p-value of the test statistic [106, 149], as $n \to \infty$, tends to

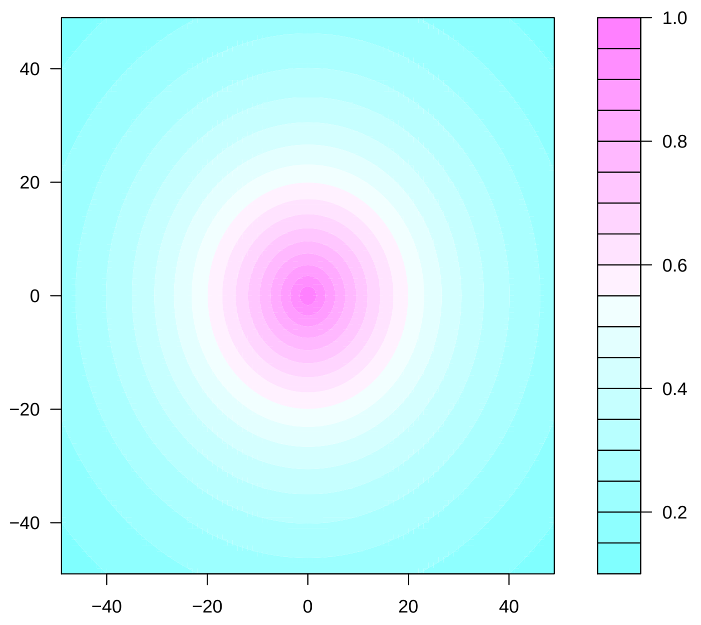

Figure 6.2.2: Heat-Plot of the isotropic kernel with the same κ as the anisotropic kernel

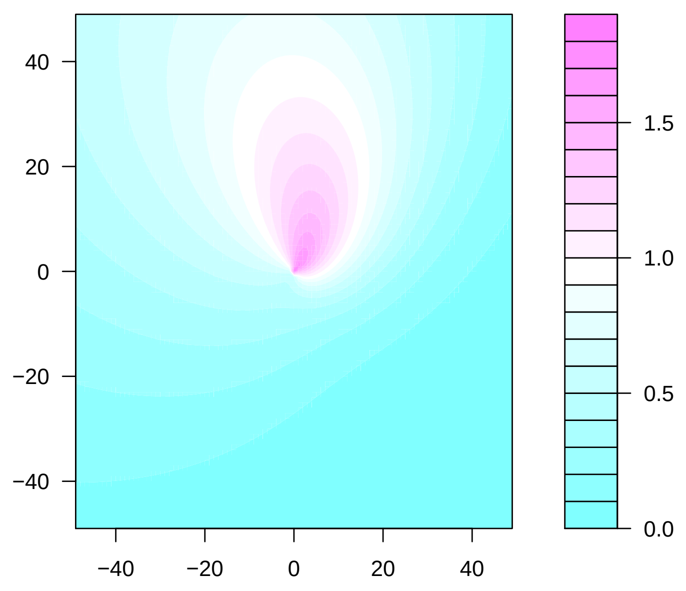

Figure 6.2.3: Heat-Plot of the anisotropic kernel given by Equation 6.2.4

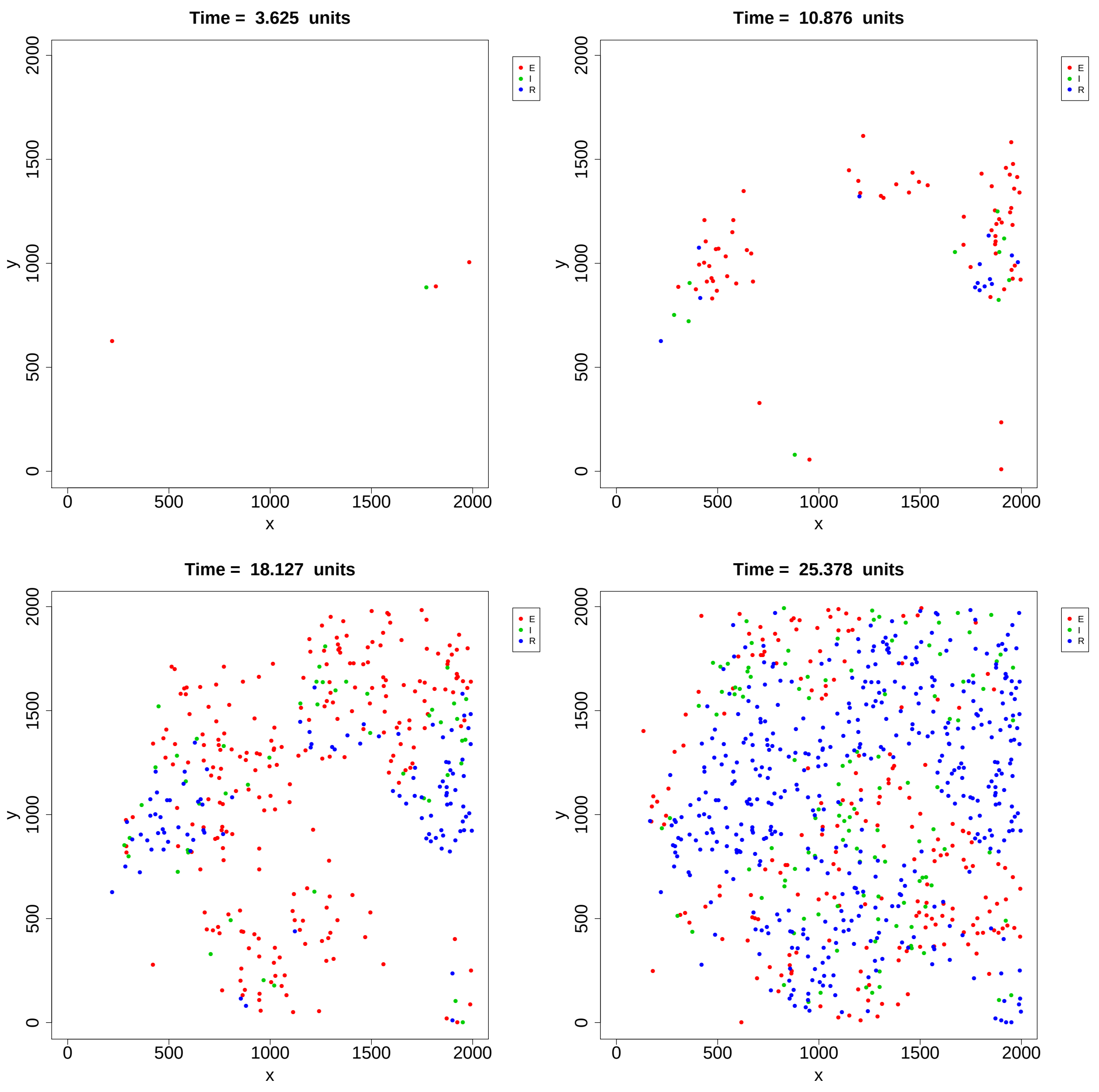

Figure 6.2.4: Snapshots of the simulated epidemic generated with the anisotropic exponential kernel in equation 6.2.3. Each point on the graph represents one host. Points are colour-coded to represent the current state of the host. Susceptible points are not displayed to maintain clarity of the graph. The colour of the points on the graph indicate the state of each host at the given time. Red indicates the host is exposed, green indicates the host is infectious and blue indicates that the host is removed.

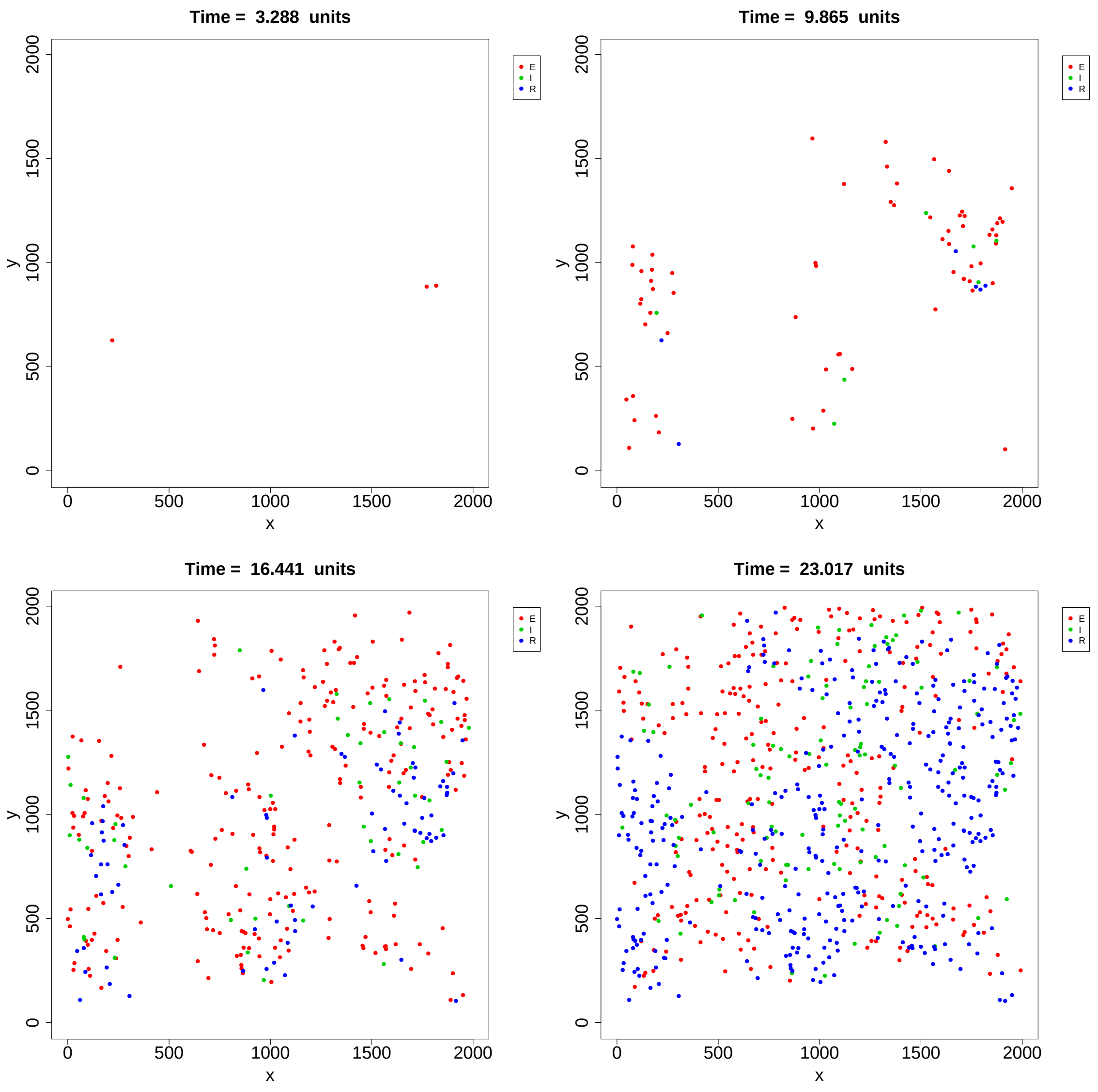

Figure 6.2.5: Snapshots of the simulated epidemic generated with the anisotropic exponential kernel in equation 6.2.4. Each point on the graph represents one host. Points are colour-coded to represent the current state of the host. Susceptible points are not displayed to maintain clarity of the graph. The colour of the points on the graph indicate the state of each host at the given time. Red indicates the host is exposed, green indicates the host is infectious and blue indicates that the host is removed.

Anisotropic kernel tested	Residuals sorted by	p-value (A-D)		p-value (Kuiper)		Parameter Updates (MCMC)	Iterations (Embedded Test)
		Mean	SD	Mean	SD		
Eq. 6.2.3	Cosine of Angle	1.104×10^{-6}	1.342×10^{-5}	1.339×10^{-6}	1.335×10^{-5}	6.6×10^{7}	804
Eq. 6.2.3	Angle	2.900×10^{-4}	1.032×10^{-3}	3.604×10^{-6}	6.02×10^{-5}	1.26×10^{7}	1135
Eq. 6.2.4	Cosine of Angle	0.049022	0.1071651	0.151480	0.1980555	1.36×10^{7}	1201
Eq. 6.2.4	Angle	8.347×10^{-7}	5.828×10^{-6}	1.718×10^{-8}	3.599×10^{-7}	1.4×10^{7}	1247

Table 6.1: Results of runs to determine the effectiveness of the directional ILR: details of anisotropic kernel used to generate the data, method of sorting the infection link residual during the sorting stage of the calculation, summary statistics of posterior latent p-values given by the Anderson-Darling and Kuiper test and iterations performed.

Anisotropic kernel tested	Residuals sorted by	p-value percentile (A-D)				
		2.5%	25%	50%	75%	97.5%
Eq. 6.2.3	Cosine of Angle	6.16×10^{-7}	6.205×10^{-7}	6.224×10^{-7}	6.250×10^{-7}	6.316×10^{-7}
Eq. 6.2.3	Angle	6.195×10^{-7}	1.125×10^{-6}	2.028×10^{-5}	1.474×10^{-4}	2.503×10^{-3}
Eq. 6.2.4	Cosine of Angle	7.42×10^{-6}	0.001112	0.008110	0.40948	0.399922
Eq. 6.2.4	Angle	6.304×10^{-7}	6.349×10^{-7}	6.376×10^{-7}	6.410×10^{-7}	6.478×10^{-7}

Table 6.2: Results of runs to determine the effectiveness of the directional ILR: details of anisotropic kernel used to generate the data, method of sorting the infection link residual during the sorting stage of the calculation, percentiles of posterior latent p-values given by the Anderson-Darling test.

$$p = Q\left(\left[\sqrt{n} + 0.155 + \frac{0.24}{\sqrt{n}}\right] V\right)$$

where

$$Q(\lambda) = 2\sum_{j=1}^{\infty}(4j^2\lambda^2 - 1)e^{-2j^2\lambda^2}$$

$$Q(0) = 1$$

$$Q(\infty) = 0$$

This power series is used to calculate the p-value in the computer code.

The Kuiper test is often used in circular statistics to test goodness of fit for a sample of random angles, against a null hypothesised distribution as the Kuiper test is rotation invariant, which is, any angular rotational shift of the data will produce the same p-value. It is hoped that the Kuiper test will therefore be sensitive to clustering of imputed infection link residuals at any section of the unit interval, unlike the Anderson-Darling test.

Anisotropic kernel tested	Residuals sorted by	p-value percentile (Kuiper)				
		2.5%	25%	50%	75%	97.5%
Eq. 6.2.3	Cosine of Angle	1.355×10^{-17}	1.171×10^{-12}	7.725×10^{-11}	7.818×10^{-9}	5.005×10^{-6}
Eq. 6.2.3	Angle	5.880×10^{-17}	9.282×10^{-13}	1.076×10^{-10}	7.636×10^{-9}	7.455×10^{-6}
Eq. 6.2.4	Cosine of Angle	2.753×10^{-4}	0.016117	0.065929	0.202866	0.695858
Eq. 6.2.4	Angle	4.008×10^{-21}	3.056×10^{-16}	3.699×10^{-14}	5.158×10^{-12}	1.302×10^{-8}

Table 6.3: Results of runs to determine the effectiveness of the directional ILR: details of anisotropic kernel used to generate the data, method of sorting the infection link residual during the sorting stage of the calculation, percentiles of posterior latent p-values given by the Kuiper test.

6.3 Results and discussion

See tables 6.1, 6.2, 6.3 for the results of the four runs performed. An over-conservative burn-in was determined visually. The first run in the table shows that the test is able to detect anisotropy in data generated using kernel 6.2.3. When the kernel used for data generation is switched to kernel 6.2.4 there is an increase in the expected posterior predictive p-value given by the test. An additional run was performed using the angle of the SI links to sort the residuals (as opposed to the cosine of the angle) yielding a lower mean latent p-value, suggesting that this form of the test was more able to detect the anisotropy better. However, when this version of the DILR test was applied to data generated from kernel 6.2.3, a larger mean latent p-value was obtained. Further investigation is needed to determine the circumstances in which sorting method outperforms the other.

Both p-values provided by the Anderson-Darling test (see Section 4.2.4.3.2) and the Kuiper tests [106] (p-value calculated with approximate distribution for large samples from [149]) were similar, however, readers should be reminded that the p-values calculated are approximate, and from power series. The approximate Anderson-Darling test statistic distribution was obtained in [116]. The approximate distribution consists of the partial evaluation of the power series (for large sample sizes), with small sample corrections applied using splines that were fitted to the approximate distributions of the test statistic via Monte-Carlo with simulated data. The p-values obtained are thought to be accurate to 6 decimal places. The Kuiper test uses a large sample approximation of the p-value, with the partial evaluation of the power series used to obtain the approximate p-value. It is thought that since the size of the host population is 1000, the large sample approximation is justified.

Success at this stage motivates the case study below on a real-world data-set.

The functional-model representation of the epidemic model can allow the creation of test statistics which focus on certain aspects of discrepancy. An interesting question or extension to this analysis is the question of the derivation of optimal test statistics for the testing of certain aspects of discrepancy. One potential area which is worth examination is the use of utility-based methods to find optimal test statistics. This can be used not only to focus tests on various areas of discrepancy but for example determine the optimal times to observe the epidemic, or the optimal hosts to include in the test.

The ease of adaption of the infection link residual test to directional infection link residuals test shows the ability of the functional-model representation and latent residuals approach to devise tests which are able to target various aspects of model mis-specification.

The use of the angle as a method of ordering the infection links appears more able to reveal discrepancy than the cosine of the angle. This is possibly because the cosine of the angle is less informative than the angle – since $\cos(\phi) = \cos(2\pi - \phi)$.

Consider as an extreme example: suppose that the actual kernel is such that there is no force of infection to the right of an infected host and the test direction vector is set to an unit vector pointing north. The epidemic will progress to the left as time passes. When the infection links are imputed, there will be a surplus of infection links to the right. Since $\cos(\phi)$ does not differentiate between angles clockwise or anticlockwise from the test direction vector there would be no concentration of infection link residuals at any single interval in $[0, 1]$. If the angle were used to order the infection links, there would be a concentration of residuals close to 0. Thus, mis-specification would be a lot harder to detect with the infection links ordered by the cosine of the angle instead of ordering by the angle.

Thus, care must be taken when specifying the latent residuals. mis-specification of the latent residuals can lead to a lack of power in the test. Moreover, the implication of this finding for non-directional infection link residual tests is that tests which order the infection links by the order of a non-invertible transformation may lead to a loss of power in tests. For example, suppose that the infection links are ordered by the size of the difference from a fixed value, and this fixed value happens to coincide with the median value of all the infection links. Since this ordering does not take into account where the infection link is in the head of the distribution or the tail of the distribution, the test would not be able to detect evidence of misfit. Hence, whilst the choice of ordering can focus the mis-specification making the test more powerful in some circumstances, there is a danger that the choice of ordering could obfuscate trends which indicate mis-specification.

Chapter 7

Case study: the Foot and Mouth Disease (FMD) outbreak of 2001

The foot and mouth disease (FMD) outbreak of 2001 involved approximately 2000 confirmed cases of FMD, where each "case" represents an infected premises. In addition, approximately 6,000,000 sheep, cattle and pigs were slaughtered to prevent the infection from spreading, with pyres and large burial pits (for the disposal of culled animal carcasses) being seen across the British countryside. This cost the UK public sector approximately £3 billion and the private sector approximately £5 billion [13, 15, 6].

The first cases of Foot and Mouth disease were discovered by routine inspection at the Cheale Meats abattoir on February 19, 2001 in which 27 pigs were found to have symptoms of FMD [13, 15, 6]. A movement ban on livestock from areas affected by FMD (and business and other livestock linked with these areas) [37, 15] and was put in place on 23 February 2001, and culling of infected animals was started on 24th of February 2001 [13, 15, 6].

The last recorded case of the outbreak was found on September 30, 2001 and by 14th of January 2002 the United Kingdom declared itself free of FMD [13, 15, 6].

There were several facets to the culling strategy, put in place as the existing measures were found to be unable to stop the epidemic [182, 6]:

1. Susceptible animals on premises on which there were clinically confirmed cases of foot-and-mouth disease were to be culled within 24 hours of report.

2. All susceptible animals on premises contiguous to any of these infected premises were to be culled within 48 hours.

3. In Cumbria, Dumfries and Galloway, sheep, pigs and goats that were within 3 kilometres of infected premises were culled.

4. Slaughter on suspicion – the culling of premises regardless of whether or not there was clinical confirmation. If a vet suspected a premises was infected with FMD but there was not enough evidence to clinically classify the case as FMD, all susceptible animals on the premises were slaughtered [6]. If serological testing confirmed that the case was indeed FMD, the premises was reclassified as an infected premises [182].

Of particular note was the use of mathematical models to inform the policy decisions to control the disease: these began with Sir John Krebs, chairman of the Food Standards Agency meeting with epidemic modelling experts to obtain predictions and analysis on the epidemic. This eventually led to the formation of the FMD Science group which met 31 times from 26th March to the 1st October [6]. This consisted of several groups of the modellers [100, 49, 48, 129] working independently of each other. The results from these models were taken into account when deciding whether to make changes to the control strategy [6].

In this thesis, we present as a case study of how our model selection techniques work in the setting of a real-world epidemic by analysing a data-set from DEFRA of the 2001 FMD outbreak. This contains 188,361 plots of land, the coordinates of the farmhouse belonging to each plot of land, the times of notification and removal of infected animals, and the number of animals of each species at each site. This was a collaboration with Chris Jewell of Lancaster University, who had fitted a model (described below) to the data set using MCMC and who provided 3000 MCMC samples from the posterior distribution which was used in our analysis.

The model fitted by Jewell (in [150]) to the data consists of four states: susceptible S, infected I, notified N, removed R. Farms which move onto the next stage of infection cannot move backwards and recover. Farms which are in the notified state and the infected state can infect farms in the susceptible state.

There is a period in which farms are infected but not infective, which is assumed to be fixed four days in Jewell's model. Let $\lambda(t)$ be the infectious pressure on susceptible j at time t. Then

$$\lambda(j) = \epsilon(t) + \sum_{i \in I(t)} \beta_{ij} h(t_j^I - t_i^I) + \sum_{i \in N(t)} \beta_{ij}^* h(t_j^I - t_i^I)$$

where

$$h(t) = \begin{cases} 0 & t < 4 \text{ days} \\ 1 & \text{otherwise} \end{cases}$$

The "movement ban" is modelled by Jewell altering the primary infection rate [150] such that

$$\epsilon(t) = \begin{cases} \epsilon_1 & t < 23 \text{ days} \\ \epsilon_1 \epsilon_2 & \text{otherwise} \end{cases}$$

The transmission kernel used consists of a susceptibility term, an infectivity term and a distance dependency term.

For $i \in I(t), j \in S(t)$, the force of infection exerted on premises j from infected premises i is modelled as:

$$\beta_{ij} = \gamma_1 \left[\left(\frac{c_i}{\bar{c}}\right)^{\psi_1} + \xi_2 \left(\frac{p_i}{\bar{p}}\right)^{\psi_2} + \xi_3 \left(\frac{s_i}{\bar{s}}\right)^{\psi_3} \right] \left[\left(\frac{c_j}{\bar{c}}\right)^{\phi_1} + \zeta_2 \left(\frac{p_j}{\bar{p}}\right)^{\phi_2} + \zeta_3 \left(\frac{s_j}{\bar{s}}\right)^{\phi_3} \right] \left[\frac{\delta}{(\delta^2 + \rho_{ij}^2)^{\omega}} \right]$$

where c_i, s_i, p_i are the numbers of cattle, pigs and sheep at site i, $\bar{c}, \bar{s}, \bar{p}$ are the mean numbers of cattle, pigs and sheep at each site.

For $i \in N(t), j \in S(t)$, the force of infection exerted on premises j from infected and notified premises i is modelled as:

$$\beta_{ij}^* = \gamma_2 \beta_{ij}$$

Here the susceptibility $\beta^{(S)}$, infectivity $\beta^{(I)}$ and distance dependency terms $\beta^{(D)}$ are:

$$\begin{aligned}\beta_i^{(I)} &= \left(\frac{c_i}{\bar{c}}\right)^{\psi_1} + \xi_2\left(\frac{p_i}{\bar{p}}\right)^{\psi_2} + \xi_3\left(\frac{s_i}{\bar{s}}\right)^{\psi_3}\\ \beta_j^{(S)} &= \left(\frac{c_j}{\bar{c}}\right)^{\phi_1} + \zeta_2\left(\frac{p_j}{\bar{p}}\right)^{\phi_2} + \zeta_3\left(\frac{s_j}{\bar{s}}\right)^{\phi_3}\\ \beta_{ij}^{(D)} &= \frac{\delta}{(\delta^2+\rho_{ij}^2)^{\omega}}\end{aligned}$$

and hence:

$$\begin{aligned}\beta_{ij} &= \gamma_1\beta_i^{(I)}\beta_j^{(S)}\beta_{ij}^{(D)}\\ \beta_{ij}^{*} &= \gamma_2\beta_{ij}\end{aligned}$$

As noted in Chapter 5, section 5.8, page 148, because of computer memory limitations, a truncation was applied to the kernel, in effect changing the spatial kernel to:

$$\beta_{ij}^{(D)} = \begin{cases}\frac{\delta}{(\delta^2+\rho_{ij}^2)^{\omega}} & \rho_{ij} < 25\text{km}\\ 0 & \text{Otherwise.}\end{cases}$$

The sojourn times in the I state were modelled as being independently gamma distributed:

$$t_N^{(i)} - t_I^{(i)} \sim \text{Gamma}(4, b)$$

7.1 Methodology

The MCMC for the parameter posterior distribution was carried out on the FMD 2001 epidemic data-set by Jewell [150]. The MCMC was run for 100,000 iterations, 40,000 iterations were discarded as burn-in and the output was thinned by 20 iterations to yield 3,000 draws from the posterior distribution of the parameters. These data were then provided to the project to be used as input to our model assessment methods. The data was comprised of:

- The County-Parish-Holding (CPH) number of each farm, the coordinates of each farm house (northing and easting), the number of cattle, sheep, pigs and deer on each farm.
- The times of notification and removal for each CPH.
- The 3000 MCMC draws from the posterior distribution $\pi(\theta, z|y)$. The augmented data z consists of the unobserved infection times $t_I^{(1)}, t_I^{(2)}, \dots t_I^{(188361)}$. The parameters are $\theta = (\epsilon_1, \epsilon_2, \gamma_1, \gamma_2, \xi_2, \xi_3, \psi_1, \psi_2, \psi_3, \zeta_2, \zeta_3, \phi_1, \phi_2, \phi_3, \delta, \omega, b)$. Each draw consists of an MCMC sample of θ and z.

Code was written to read the data sent by Jewell [94] into the C++ program. The data-set posed challenges regarding memory usage because of its large size. Extensive modification was needed the code to adapt the routines to run on the data-set. Eventually the infection link residuals routine, which was previously written as a program for the CPU had to be ported to the GPU (see Chapter 5 for details of the implementation and the challenges involved), as the CPU program took one hour per p-value calculation and would often cause the computer to crash. The GPU program takes approximately 1 minute to calculate the p-value of the ILR test from a single draw from the posterior distribution that had been read into the program (on a consumer grade NVIDIA GTX Titan graphics card) and uses almost all of the available GPU memory and most of the CPU memory.

The ILR test was performed on the MCMC output supplied by Jewell. This shows one of the advantages of the embedded testing methods which have been developed: once the MCMC has been run, the process of model testing can be done at a later time provided that the values of the parameters and unobserved data from the MCMC have been saved. Two candidate models were fitted to the data (results from run with Cauchy kernel were used in [150]). The first model fitted used a Cauchy kernel for the distance dependency term [150]:

	Mean	SD
ϵ_1	1.443×10^{-6}	1.472×10^{-7}
ϵ_2	1.052×10^{-7}	7.748×10^{-8}
γ_1	6.367×10^{-3}	1.018×10^{-3}
γ_2	3.446×10^{-1}	5.675×10^{-2}
ξ_2	3.530×10^{-1}	3.039×10^{-1}
ξ_3	8.812×10^{-1}	2.460×10^{-1}
ψ_1	3.251×10^{-1}	1.082×10^{-1}
ψ_2	4.246×10^{-1}	2.051×10^{-1}
ψ_3	1.095×10^{-1}	7.021×10^{-2}
ζ_2	6.842×10^{-2}	3.580×10^{-2}
ζ_3	1.035×10^{0}	8.836×10^{-2}
ϕ_1	7.383×10^{-1}	3.716×10^{-2}
ϕ_2	5.838×10^{-1}	1.271×10^{-1}
ϕ_3	4.706×10^{-1}	3.030×10^{-2}
δ	1.266×10^{0}	6.077×10^{-2}
ω	1.300×10^{0}	0.000×10^{0}
b	4.618×10^{-1}	1.615×10^{-2}

Table 7.1: Posterior summary statistics from FMD2001 run with the Cauchy kernel obtained by Jewell [94].

	Mean	SD
ϵ_1	2.184×10^{-6}	2.268×10^{-7}
ϵ_2	1.0	0.0
γ_1	1.733×10^{-3}	3.012×10^{-4}
γ_2	1.0	0.0
ξ_2	3.443×10^{-1}	3.154×10^{-1}
ξ_3	9.197×10^{-1}	2.657×10^{-1}
ψ_1	2.954×10^{-1}	1.177×10^{-1}
ψ_2	3.948×10^{-1}	2.036×10^{-1}
ψ_3	6.909×10^{-2}	5.051×10^{-2}
ζ_2	6.429×10^{-2}	3.359×10^{-2}
ζ_3	1.007	8.622×10^{-2}
ϕ_1	7.229×10^{-1}	3.746×10^{-2}
ϕ_2	5.742×10^{-1}	1.273×10^{-1}
ϕ_3	4.673×10^{-1}	3.091×10^{-2}
δ	4.504×10^{-1}	1.415×10^{-2}
ω	1.330	0.0
b	4.965×10^{-1}	1.646×10^{-2}

Table 7.2: Posterior summary statistics from FMD2001 run with the Exponential kernel obtained by Jewell [94].

$$\beta^{(D)} = \frac{\delta}{(\delta^2 + \rho_{ij}^2)^{\omega}}$$

After truncation has been applied to the kernel (see section 5.8, page 148 for motivation), this is effectively:

$$\beta_{ij}^{(D)} = \begin{cases} \frac{\delta}{(\delta^2+\rho_{ij}^2)^{\omega}} & \rho_{ij} < 25\text{km} \\ 0 & \text{Otherwise} \end{cases}$$

The second model is one which uses an Exponential kernel for the distance dependency term:

$$\beta^{(D)} = \exp(-\delta\rho_{ij})$$

After truncation has been applied to the kernel (see section 5.8, page 148 for motivation), this is effectively:

$$\beta_{ij}^{(D)} = \begin{cases} \exp(-\delta\rho_{ij}) & \rho_{ij} < 25\text{km} \\ 0 & \text{Otherwise} \end{cases}$$

Both these kernels were evaluated for goodness of fit using the ILR and DILR test statistics. The ILR and the DILR test statistic were calculated as before in previous chapters, with slight modifications for the different model involved:

7.1.1 Calculation of Test Statistics and Posterior Latent p-values

As introduced in Section 3.3.2 on page 73, and the paper [111], recall that the definition of the infection link residuals (ILR) test is:

Definition 29 (Infection Link Residual)**.** Let

$$p_{mn} \propto \begin{cases} \beta_{nm} h(t^I_m - t^I_n) & \text{if } m \in S(t_k) \text{ and } n \in I(t_k) \\ \beta^*_{nm} h(t^I_m - t^I_n) & \text{if } m \in S(t_k) \text{ and } n \in N(t_k) \end{cases}$$

for all $m \in S(t_k)$ and $n \in I(t_k) \cup N(t_k)$.

Order the p_{mn} such that $p_{(1)} \leq p_{(2)} \leq p_{(3)} \leq p_{(4)} \leq \ldots$.

Let $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf \left\{ s | \tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)} \right\} = s'$$

The infection link residual test is calculated by the following algorithm (as mentioned in Section 3.3.2 on page 75, from [111]):

1. The infection link for the kth exposure between individuals i and j is chosen with probability p_{ij} from the possible links at time t_k. Primary infection is treated as being an infection caused by a notional infector with force of infection α.

2. The infection links are then ordered and the ranking s' of p_{ij} is determined.

3. Generate a random deviate from Unif($\sum_{l=1}^{s'-1} p_{(l)}, \sum_{l=1}^{s'} p_{(l)}$). This is the imputed infection link residual for the kth exposure.

Definition (Directional Infection link residuals)**.** Suppose that host j is infected by infectious host i and that this is the kth exposure event. The infection link residual for this infection time is defined by the following:

Let

$$p_{mn} \propto \begin{cases} \beta_{nm} h(t^I_m - t^I_n) & \text{if } m \in S(t_k) \text{ and } n \in I(t_k) \\ \beta^*_{nm} h(t^I_m - t^I_n) & \text{if } m \in S(t_k) \text{ and } n \in N(t_k) \end{cases}$$

for all $m \in S(t_k)$ and $n \in I(t_k) \cup N(t_k)$.

Order the p_{mn} such that $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the cosine of the angle between the infection link and the vector $(1,1)$.

Let $p_{(s')} = p_{ij}$.

The infection link residual of the kth exposure $\tilde{r}_{2k}$ satisfies the following equation:

$$\inf\left\{s|\tilde{r}_{2k} < \sum_{l=1}^{s} p_{(l)}\right\} = s'$$

We also used an alternative version where $p_{(1)}, p_{(2)}, p_{(3)}, p_{(4)}, \ldots$ is ordered by the **angle** between the infection link and the vector $(1,1)$.

1. The infection link for the kth exposure between individuals i and j is chosen with probability p_{ij} from the possible links at time t_k. Primary infection is treated as being an infection caused by a notional infector with force of infection α. The S-I link with this notional infector has a random angle with the vector $(1,1)$ with distribution Unif$(0, 2\pi)$.

2. The infection links are then ordered and the ranking s' of p_{ij} is determined.

3. Generate a random deviate from Unif$(\sum_{l=1}^{s'-1} p_{(l)}, \sum_{l=1}^{s'} p_{(l)})$. This is the imputed infection link residual for the kth exposure.

7.1.1.1 Calculation of Imputed P-Value

The p-value is calculated using the Anderson-Darling test [7]. This is a frequentist test of the hypotheses:

H_0 :The data has cumulative distribution function $F(x)$

H_A :The data does not have cumulative distribution function $F(x)$

The data for this test is a random sample denoted $\{X_1, X_2, \ldots, X_n\}$

Let the empirical distribution function be defined as:

$$F_n(x) = \frac{\text{number of } X_1, X_2 \ldots, X_n \text{ that are} \leq x}{n}$$

The test statistic is defined as:

$$A_n = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\ln F(X_i) + ln(1 - F(X_{n+1-i}))\right] \tag{7.1.1}$$

The Anderson-Darling test statistic can be expressed in another form, which shows that it is the integral of the weighted squared difference between the empirical distribution function and the hypothesised distribution function, multiplied by a weighting with weight concentrated towards the tails of the distribution.

$$A_n = n\int_0^1 \frac{[F_n(x) - F(x)]^2}{F(x)(1-F(x))}dF(x) \tag{7.1.2}$$

This makes the Anderson-Darling test more able to detect discrepancy between the hypothesised distribution and the data and the tails of the distribution than the Kolmogorov-Smirnov test which is more commonly used.

To obtain the test statistic in Equation 7.1.1, use partial fractions on Equation 7.1.2:

$$A_n = n\left(\int_0^1 \frac{[F_n(x) - F(x)]^2}{F(x)}dF(x) + \int_0^1 \frac{[F_n(x) - F(x)]^2}{(1-F(x))}dF(x)\right) \tag{7.1.3}$$

Since the empirical CDF $F_n(x)$ is a step function, it is straightforward to obtain the test statistic given in Equation 7.1.1.

In this case $F(x)$ is an uniform cdf between 0 and 1, the hypotheses and test statistics are:

H_0 :The data has cumulative distribution function $F(x)$

H_A :The data does not have cumulative distribution function $F(x)$

The test statistic is simplified to:

$$A_n = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left[\ln X_i + ln(1 - X_{n+1-i})\right] \tag{7.1.4}$$

Regarding the derivation of the test statistic, start with:

$$A_n = n\int_0^1 \frac{[F_n(x)-x]^2}{x(1-x)}dx$$

Use partial fractions on the integral to obtain:

$$A_n = n\left(\int_0^1 \frac{[F_n(x)-x]^2}{x}dx + \int_0^1 \frac{[F_n(x)-x]^2}{(1-x)}dx\right)$$

Since the empirical distribution function $F_n(x)$ of the data is a step function, it is straightforward to integrate and simplify to obtain the test statistic in Equation 7.1.4.

The Anderson-Darling test is performed upon the infection link residuals that are obtained through the algorithm described on page 196. The algorithm used to calculate the p-value is the algorithm described in [116], which is an approximation accurate up to 6 decimal places.

7.2 Results

Tables 7.3 on page 201 and 7.4 show the posterior means and percentiles for the posterior distribution for the p-value of the infection link residuals test statistic using the Anderson-Darling test for uniformity. As can be seen from the tables, tests of both kernels produced small posteriors means and percentiles, indicating that there is some significant discrepancy between the data and both fitted models, as almost all of both posterior distributions have most of their mass below 0.01. There appears to be smaller p-values for the exponential distribution but this is only because the distribution appears to be longer tailed to right, perhaps because some of the infections can be explained by the larger mean posterior primary infection rate in the fitted exponential kernel model (see Table 7.1 and Table 7.2). Readers should recall that the primary infections are treated as being infected by a notional infector with force of infection equal to the force of primary infection. Further insight can be gained by looking at the infection links for each RJMCMC iteration. Fig. 7.2.1 and Fig. 7.2.2 show two-dimensional histograms of the residuals for each MCMC iterations with the iteration number on the y-axis, the x-axis showing the intervals of each bin in the histogram where the colour of each cell indicates the numbers of infection link residuals that fall within these bins. For both plots there appears to be disproportionately many of residuals at the lower end. This implies the size of the force of infection along the infection links are less than would be expected under the null hypothesis, and hence implies that the distance dependency term is too small (it is unlikely that such a bias in the infection link residuals is due to mis-specification of the susceptibility and infectivity terms, as this would produce different patterns of nonuniformity), and hence evidence that the kernel is too short tailed. This bias to smaller values is less apparent for the exponential kernel, perhaps because a larger mean posterior primary infection rate (see Table 7.1 and Table7.2) can explain some of the longer range interactions.

Likewise, the posterior means and percentiles can be found in tables 7.5 and 7.6 for the directional infection link residuals (DILR) test. Here the links have been sorted by the angle between the infection link and the vector $(1, 1)^T$. The primary infections are treated as having an infection link with a uniformly distributed random value for an angle between the vector $(1, 1)^T$ between 0 and 2π. Again, both of the models produce small p-values; there is a large amount of the posterior distribution of the latent p-value below 10% for the Cauchy kernel and the majority of the distribution is below 10 to 5%. Again, the calculations of the p-values are only approximate, the approximation being accurate to 6 decimal places, thus, not too much should be read into the very small values apart from that the very small values of p-value that have been calculated indicate the p-value is very small. Looking at the two-dimensional histograms, there appears to be some sort of systematic pattern, indicating there is a surplus of the links at certain angles. Ordering the links by the cosine of the angle yields the posterior summary

Kernel	Mean	SD
Cauchy	2.63×10^{-7}	2.26×10^{-9}
Exponential	1.63×10^{-6}	1.62×10^{-5}

Table 7.3: Infection link residuals test: posterior means and standard deviation of p-value

Kernel	2.5%	25%	50%	75%	97.5%
Cauchy	2.59×10^{-7}	2.62×10^{-7}	2.63×10^{-7}	2.65×10^{-7}	2.68×10^{-7}
Exponential	2.72×10^{-7}	2.75×10^{-7}	2.76×10^{-7}	2.78×10^{-7}	1.13×10^{-5}

Table 7.4: Infection link residuals test: posterior percentiles of p-value

statistics in tables 7.7 and 7.8. Both posterior distributions obtained have a lot of their mass below 10^{-6}. A similar systematic pattern can be observed in figures 7.2.5 and 7.2.6. Further discussion will be found on this systematic pattern below

7.3 Conclusions and Discussion

The findings of this analysis of the data show the importance of using tests which are orientated towards discrepancy relevant to the purposes of prediction. In this case, in previous work [97] testing of model adequacy was performed a similar model by considering a non-centred parametrisation on the sojourn times $t_N^{(i)} - t_I^{(i)}$. The difference between the model tested in this paper versus model tested here in this thesis is that the paper does not include pigs in the model as transmitters of the disease, since the authors of the paper considered the FMD 2001 to be mainly between cows and sheep (an assumption that may be justifiable given the susceptibility parameter for pigs, ζ_2 is substantially smaller than that for the other species). Using the non-centered parameterisation $U_i = \frac{t_N^{(i)} - t_I^{(i)}}{b}$, $U_i \sim \text{Gamma}(4, 1)$, the fit of the U_i to the Gamma(4, 1) was assessed graphically, showing no evidence of misfit. In addition, fitting Gamma(a, 1) with a as an unknown parameter yielded a posterior mode of 3.76 showing no evidence of

Kernel	Mean	SD
Cauchy	1.98×10^{-1}	2.20×10^{-1}
Exponential	5.71×10^{-2}	1.08×10^{-1}

Table 7.5: Directional Infection link residuals test (angle-based): posterior means and standard deviation of p-value

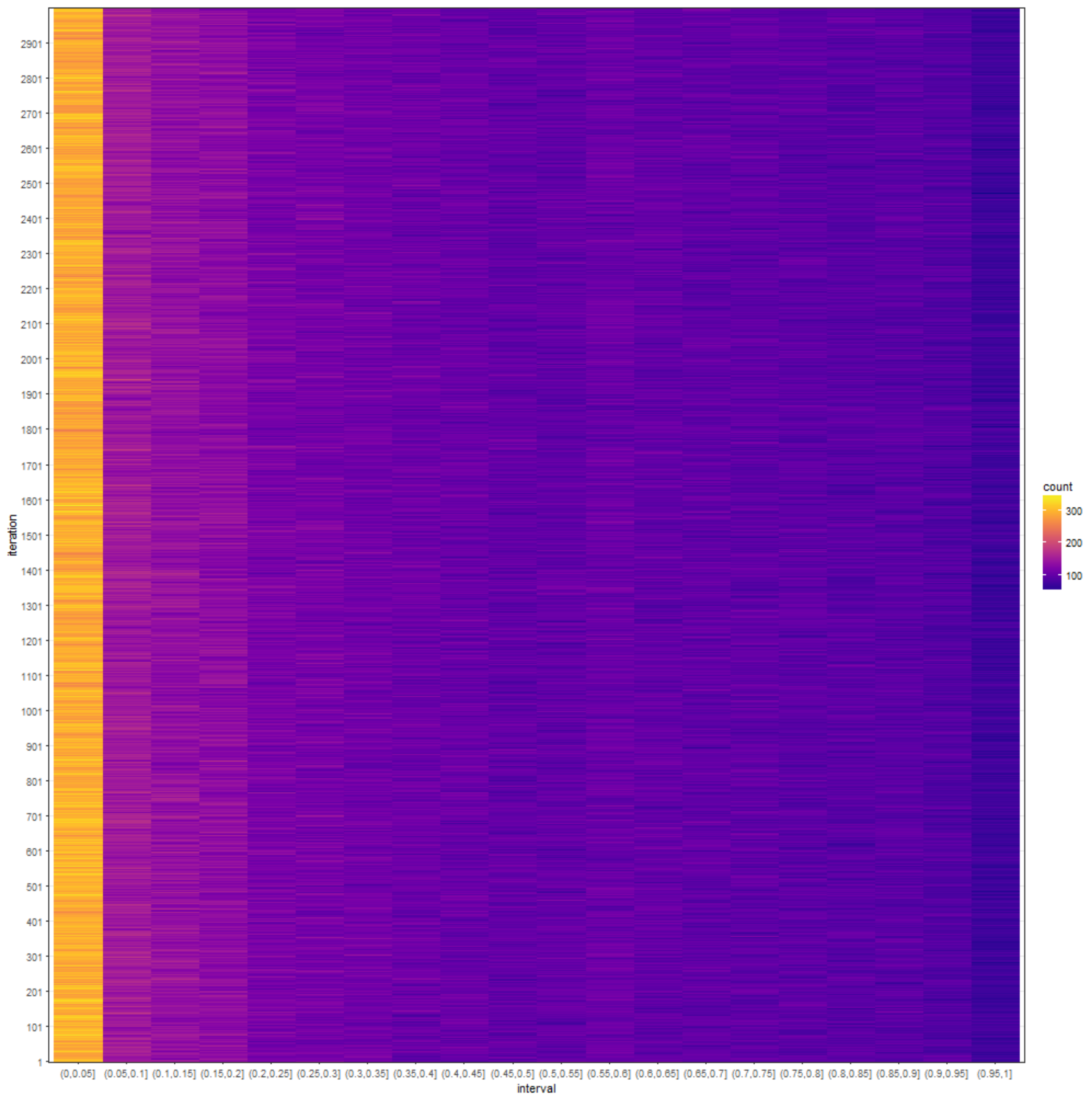

Figure 7.2.1: Infection link residuals test: 2 dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Cauchy kernel

Kernel	2.5%	25%	50%	75%	97.5%
Cauchy	1.14×10^{-3}	3.07×10^{-2}	1.09×10^{-1}	3.00×10^{-1}	7.84×10^{-1}
Exponential	2.89×10^{-5}	2.33×10^{-3}	1.29×10^{-2}	5.519×10^{-2}	3.89×10^{-1}

Table 7.6: Directional Infection link residuals test (angle based): posterior percentiles of p-value

Kernel	Mean	SD
Cauchy	2.82×10^{-7}	2.83×10^{-9}
Exponential	2.90×10^{-7}	2.15×10^{-9}

Table 7.7: Directional Infection link residuals test (cosine-of-angle-based): posterior means and standard deviation of p-value

Kernel	2.5%	25%	50%	75%	97.5%
Cauchy	2.76×10^{-7}	2.80×10^{-7}	2.82×10^{-7}	2.84×10^{-7}	2.87×10^{-7}
Exponential	2.86×10^{-7}	2.89×10^{-7}	2.90×10^{-7}	2.92×10^{-7}	2.946×10^{-7}

Table 7.8: Directional Infection link residuals test (cosine-of-angle-based): posterior percentiles of p-value

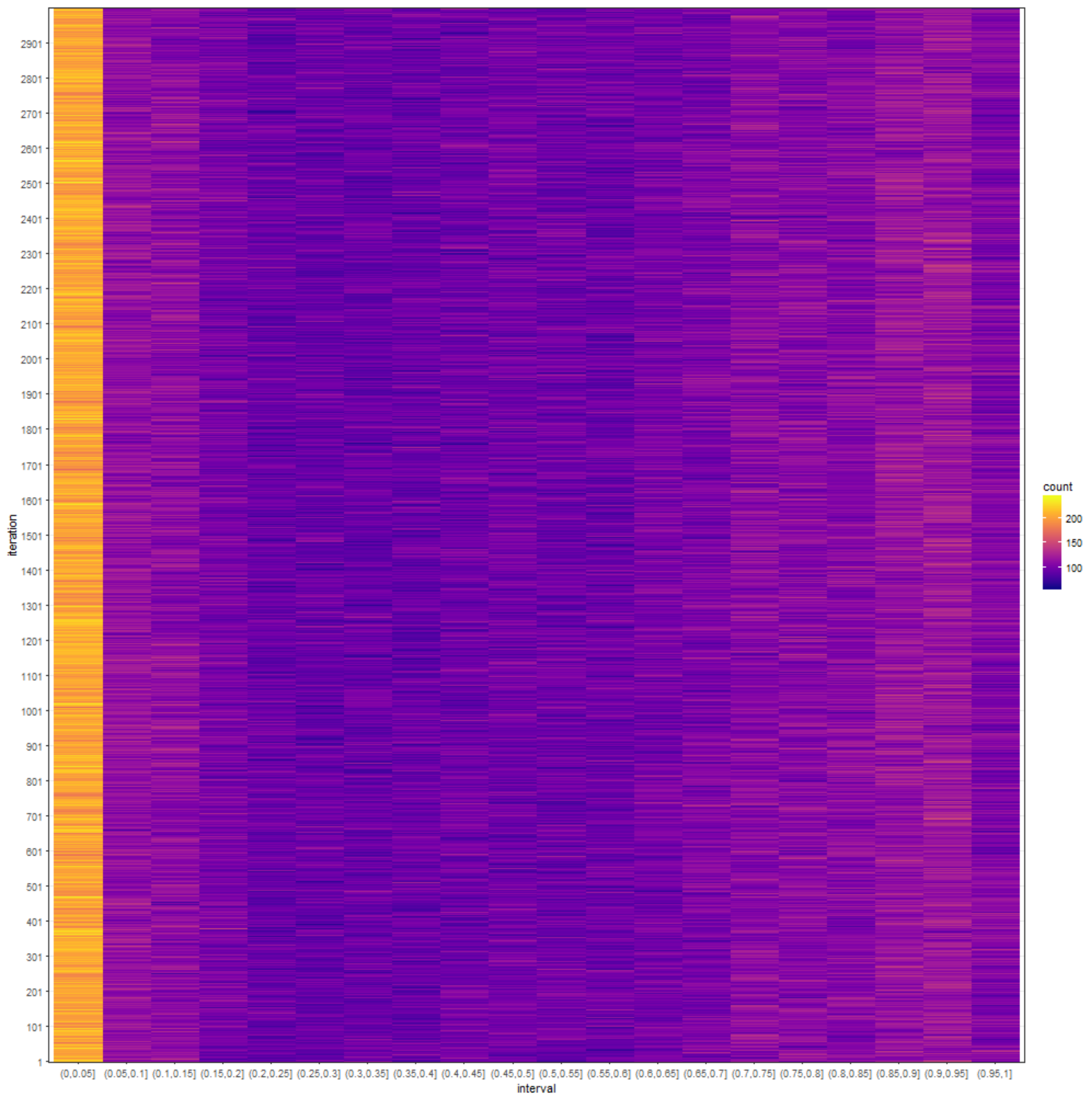

Figure 7.2.2: Infection link residuals test: 2 dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Exponential kernel

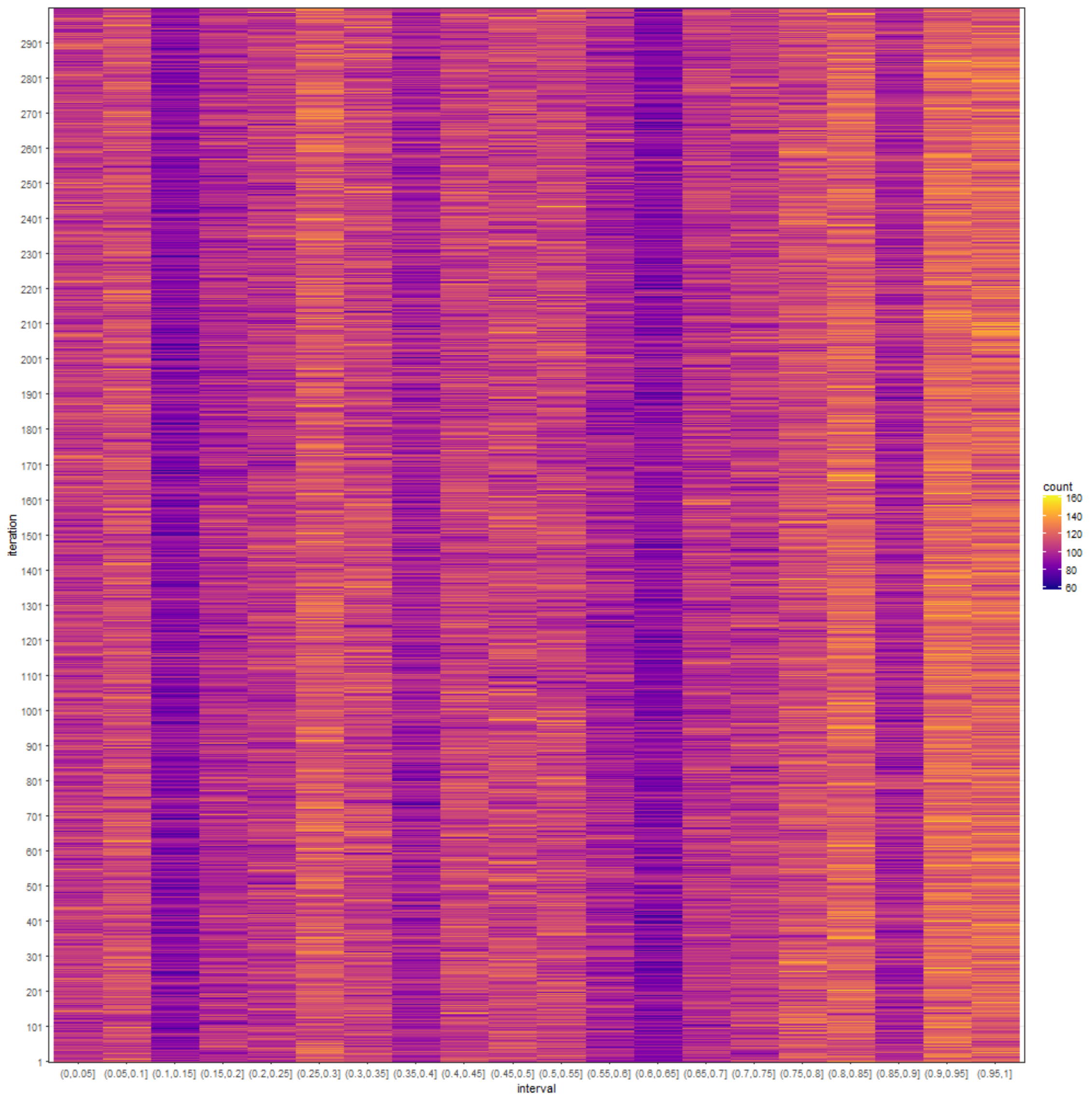

Figure 7.2.3: Directional Infection link residuals test (angle-based): 2-dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Cauchy kernel

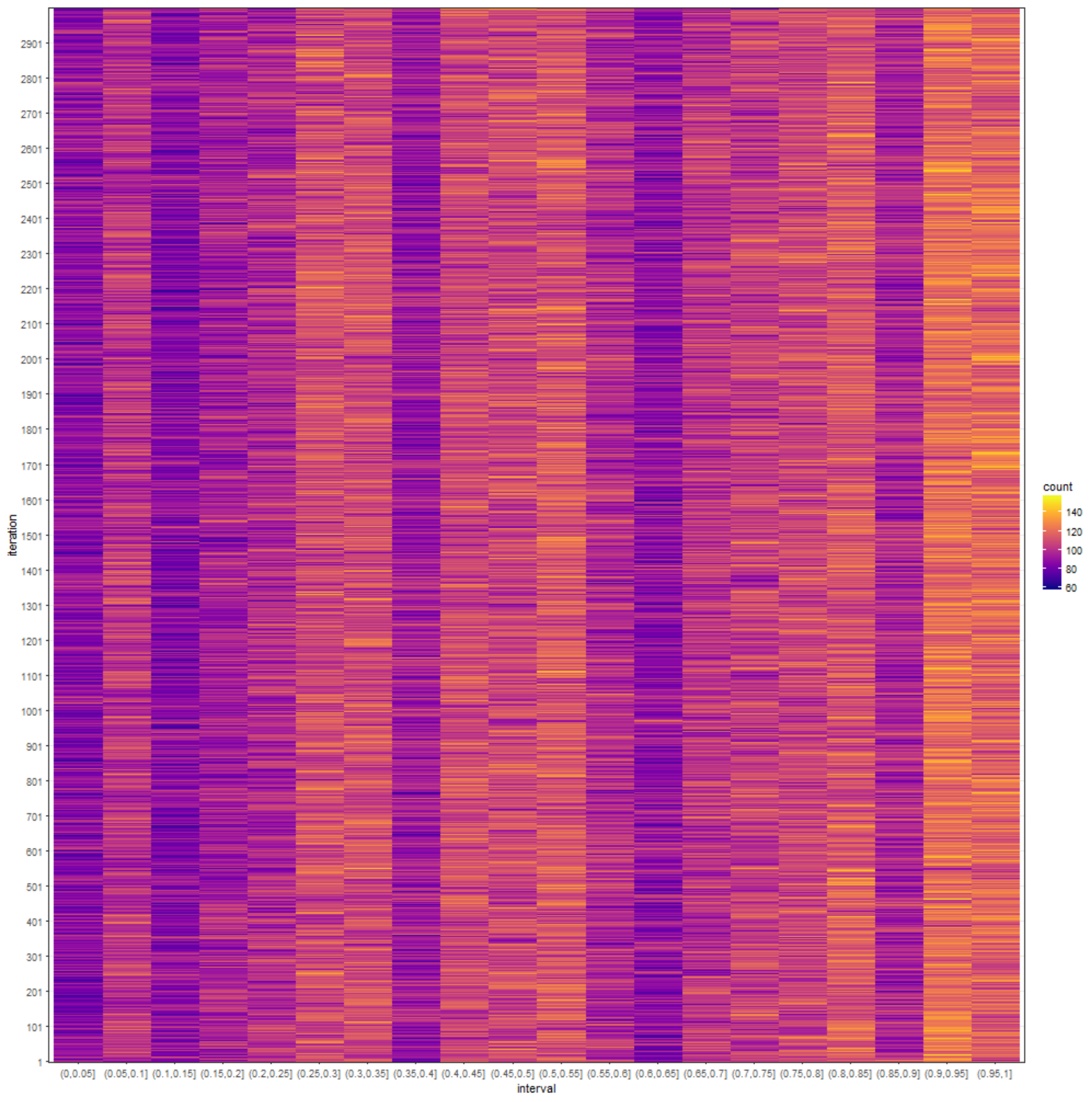

Figure 7.2.4: Directional Infection link residuals test (angle-based): 2-dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Exponential kernel

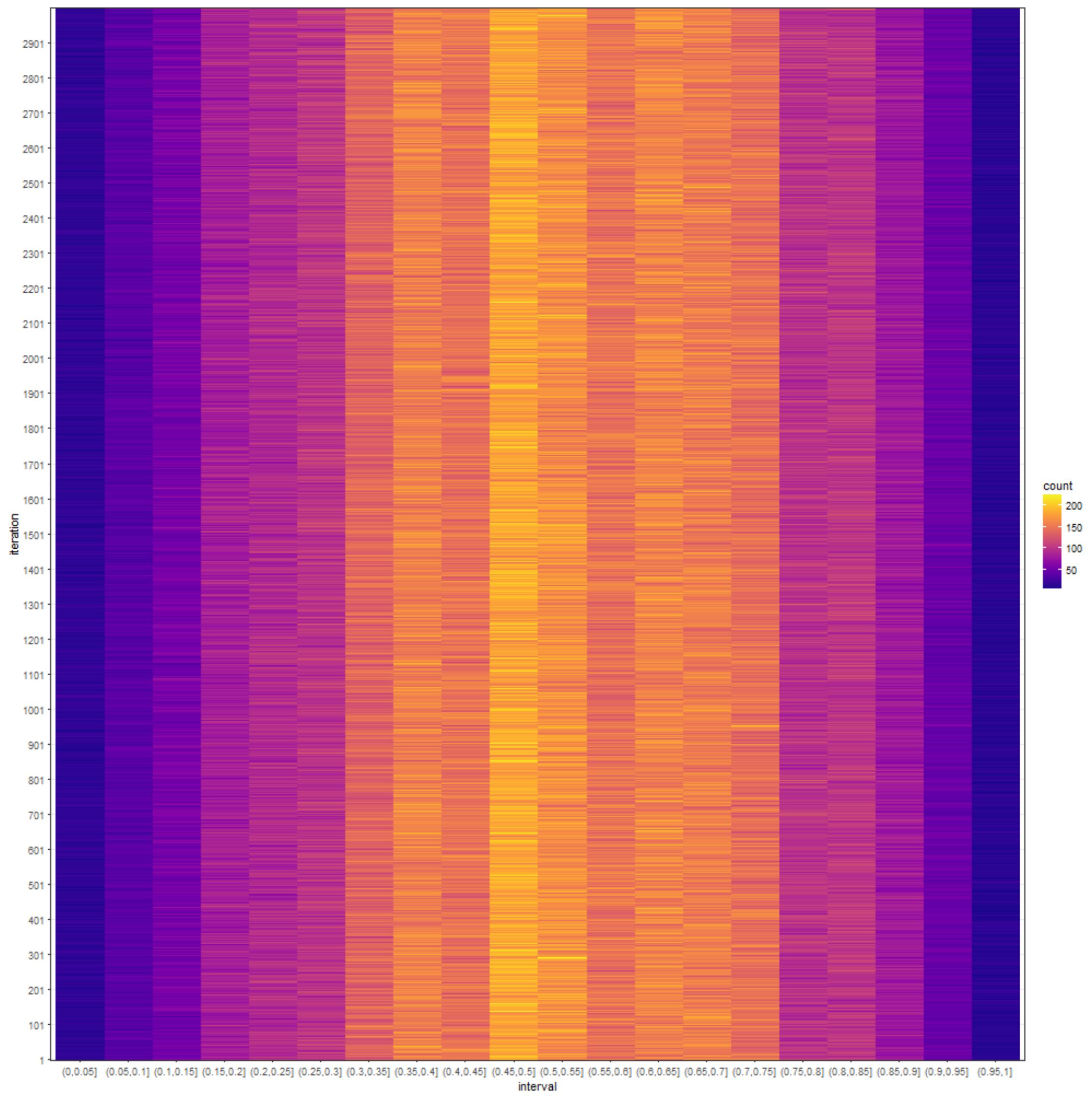

Figure 7.2.5: Directional Infection link residuals test (cosine-of-angle-based): 2-dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Cauchy kernel

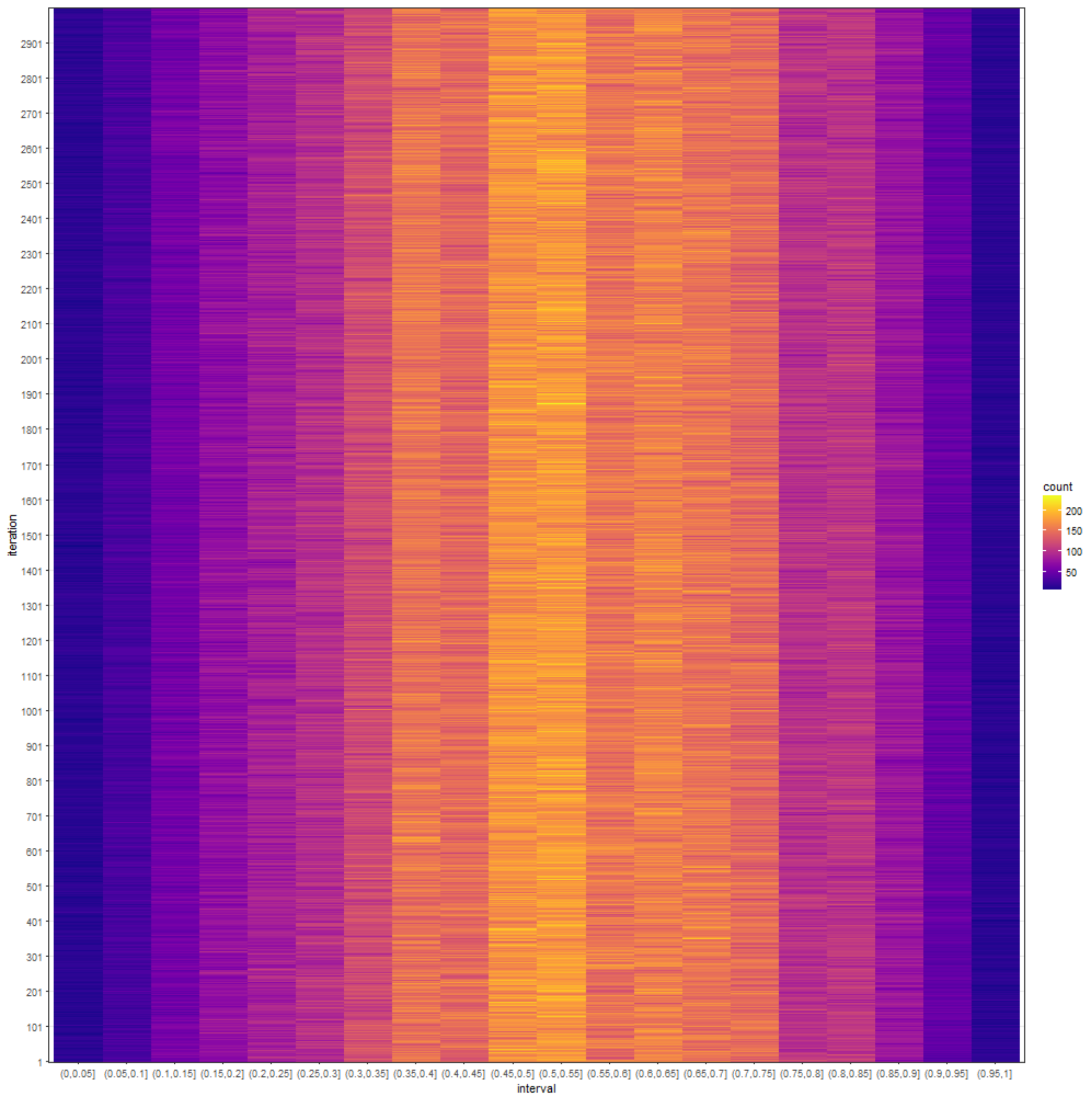

Figure 7.2.6: Directional Infection link residuals test (cosine-of-angle-based): 2-dimensional histogram of the infection link residuals at each MCMC iteration, with iteration number on the y-axis and interval on the x-axis. The kernel fitted to the data is the Exponential kernel

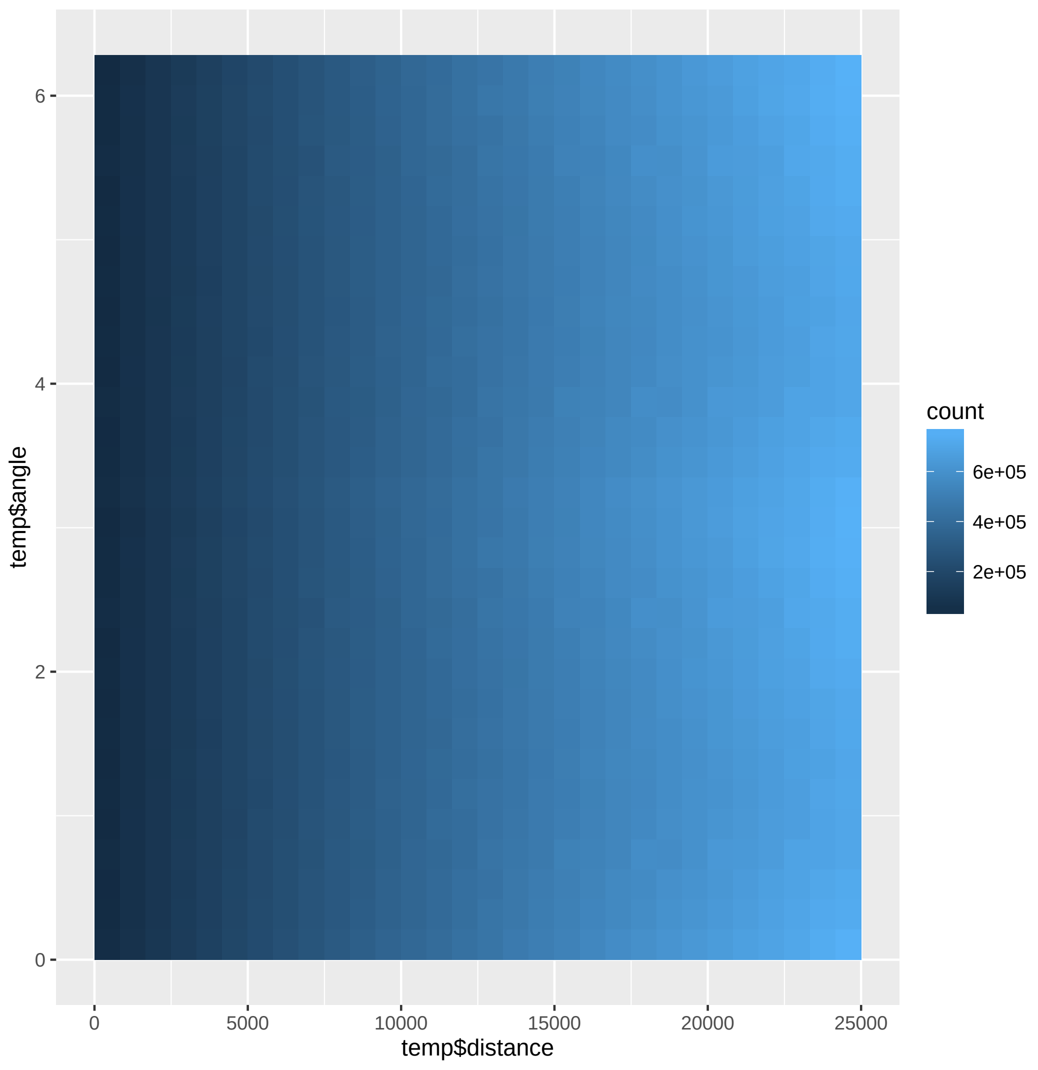

Figure 7.2.7: A plot of the angles between all farms in the FMD 2001 dataset and the vector $(1, 1)^T$

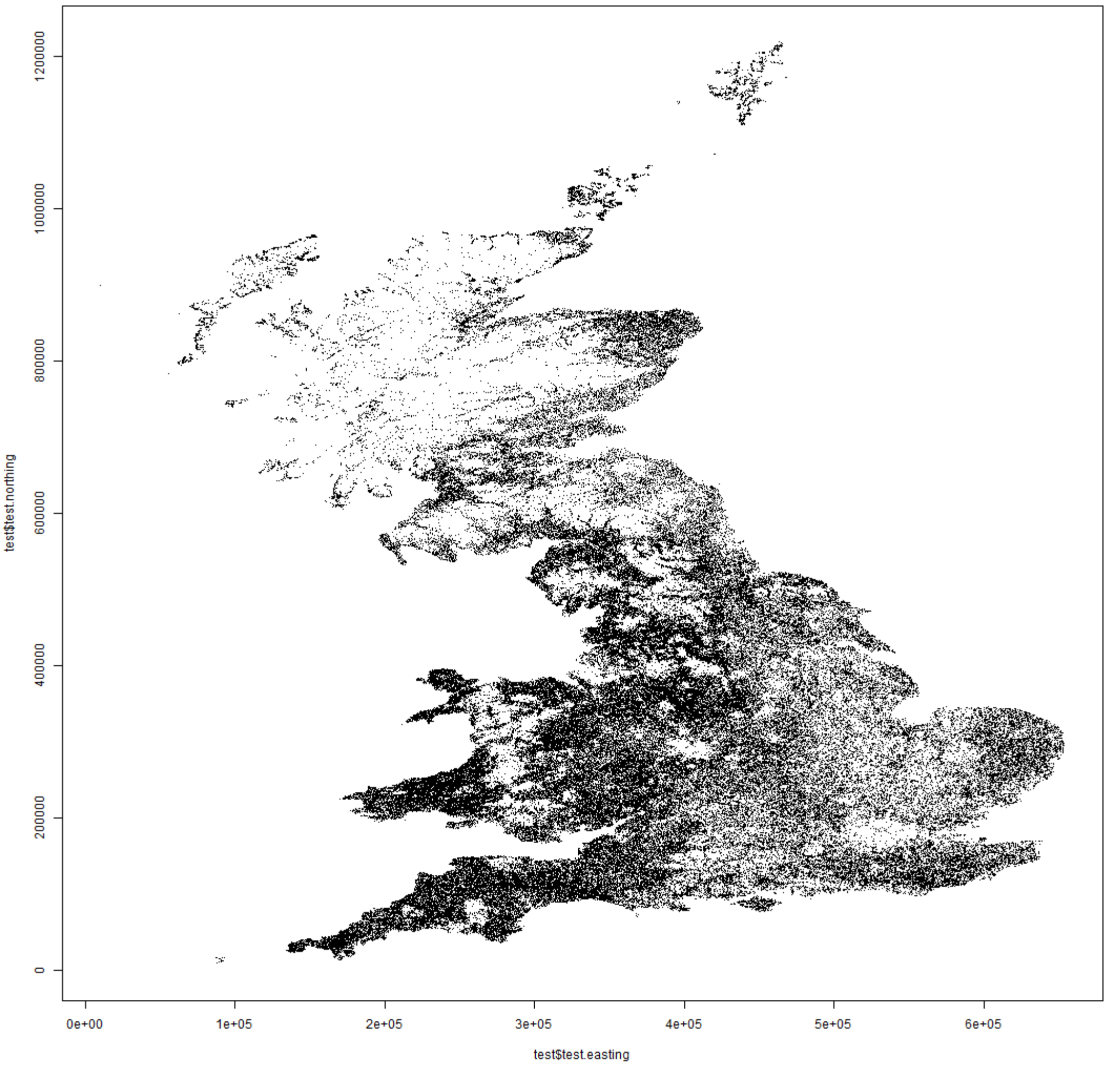

Figure 7.2.8: A plot of the position of the farms in the FMD 2001 dataset, with each point in the plot representing each farm in the dataset. These points are plotted by the northern and easting of the coordinates given in the dataset.

misfit. However, this did not test the adequacy of the tailedness of the spatial kernel, nor if there were any presence of mis-specification which would suggest the presence of anisotropy or some other mechanism which is angle based which is not explained by the model. Test that are orientated towards pertinent aspects of the model given insight into aspects of model fit which are most pertinent to the purposes of the model.

In this case study, there was significant evidence that the spatial kernel (a Cauchy kernel which was used in the original study) was mis-specified. Reference [97], when stating the reason for the choice of Cauchy kernel gives the justification "For reasons of robustness, it is prudent to adopt a heavy-tailed transmission kernel". However, the model tests performed here in this thesis, show that there is evidence to suggest that this kernel does not adequately explain the pattern of infections in the data.

Of particular interest is the fact that the tests show that neither the Cauchy kernel nor the exponential kernel appear to adequately explain the data.

This misfit may be caused by the method by which the locations of each farm have been recorded in the data-set. In the data, the locations of each farm have been recorded as the location of the farmhouse of that farm, instead of the location of the centroid of the parcel of land that constitutes that farm. Because of this, the recorded coordinates in the data are not at the centroid of each farm, leading to unintuitive items in the data, such as farms being situated at the same spatial coordinates, and therefore having zero distance between them.

Nevertheless, attempts have been made to assess how mis-specification of the coordinates affects posterior estimates. If each farmhouse is located at a random point relative to the centroid of the farmland, then the reported coordinates in the dataset can be considered to be equal to the actual coordinates of the centroid of the farmland plus some random perturbation. Hola Kwame Adrakey has been performing computer simulations on datasets of this kind [2], unpublished at the time of writing, and the results are worth mentioning here:

Simulated data were generated with known coordinates based on the Florida citrus canker data-set, which simulated data was generated with the known coordinates from the Florida citrus canker data-set, and known model and parameters. The model used was an SIR (Susceptible-Infectious-Removed) model, where the hosts in the infection are individual trees. The form of the model is effectively that of Section 2.1 on page 11, with sojourn time in the E state set to 0. The data was “observed” at snapshots 30 days apart starting at $t = 130$ days. The infection starts at $t = 0$. The hosts in the I compartment are unobserved, but are detected and removed as they enter the R compartment. The first infection was set to be that in [134].

Hosts in state S at time t, $S(t)$ experience infectious challenge from two sources: primary infectious challenge, from sources/sites external to the system under study, and secondary infectious challenge, from infectious individuals within the system. Let ϵ and β be the primary and secondary infection rates. Then the probability of exposure (for an arbitrary host $j \in S(t)$) can be modelled by the following equation:

$$\Pr(j \text{ exposed during } [t, t+dt]) = C(\boldsymbol{x}_j, t)\, dt = \left(\epsilon + \beta \sum_{i \in I(t)} K(\boldsymbol{x}_j, \boldsymbol{x}_i, \alpha)\right) dt \quad (7.3.1)$$

An exponential kernel was used to generate the data:

$$K(d, \alpha) = \frac{1}{2\pi d} \frac{1}{\alpha} \exp(-\frac{d}{\alpha})$$

The dataset was generated with parameters $\beta = 8 \times 10^{-6}\text{days}^{-1}\text{km}^2$,$\alpha = 0.8\text{km}$. There was no primary infection for the simulated data. A perturbation with distribution $N(0, \sigma^2)$ was added to each of the actual coordinates after data generation, such that the “observed” coordinates were not the actual coordinates. This was repeated several times with σ equal to 0, 0.001, 0.005, 0.01, 0.015, 0.016, 0.02, 0.03 km to generate several data-sets.

Data augmented MCMC was performed on the data to obtain parameter estimates for each of these data sets. In each computer run, the data was first generated with the known parameters and model, as stated earlier. The coordinates were then perturbed. The chain was run for 510000 iterations, with 10000 discarded as burn-in. This was then thinned to 100000 draws from the parameter posterior distribution. The posterior distribution was then estimated from this perturbed data, and then the posterior marginal distributions were then plotted to determine the effect of the perturbation variance on the posterior distributions. Figure 7.3.1 shows the posterior marginal distribution of the kernel parameter α. As the variance of the perturbation increases, the spatial kernel a posteriori is more longer-tailed.

The misfit detected in this thesis with the kernels for the FMD 2001 dataset may be caused by the following. Since all the spatial kernels fitted on the FMD 2001 data-set were truncated at 25 km, these kernels cannot fit the apparent increase (a posteriori) of long-distance interactions. Hence, there is discrepancy between the fitted model and the data, which is picked up by the infection link residuals tests.

This truncation was used to make the model fitting computationally feasible: even with modern computing power and parallel programming techniques, the data-set contains approximately 189,000 parcels of land. This creates a distance matrix with more entries than would be feasible to store in random access memory, if every entry in the matrix would be stored with an acceptable level of precision. This shows the power of the approach used in code for the earlier parts of the thesis, in which the matrix of the spatial kernel between hosts is calculated and stored, rather than the distance matrix. This allows an approach in the same vein as [29] where the spatial kernel matrix can be calculated and the entries below a certain pre-specified threshold are discarded and only entries within that thresholds are stored. This "intelligent truncation" may allow more flexibility than a plain 25 km truncation distance, allowing the kernel to be fit to the data with less discrepancy.

Another explanation is that the model does not take into account the movement of animals between the parcels of land. In the UK animals move between different areas, but this effect was not included in the model in order to lower model complexity. The lack of model fit may indicate that in order to accurately predict

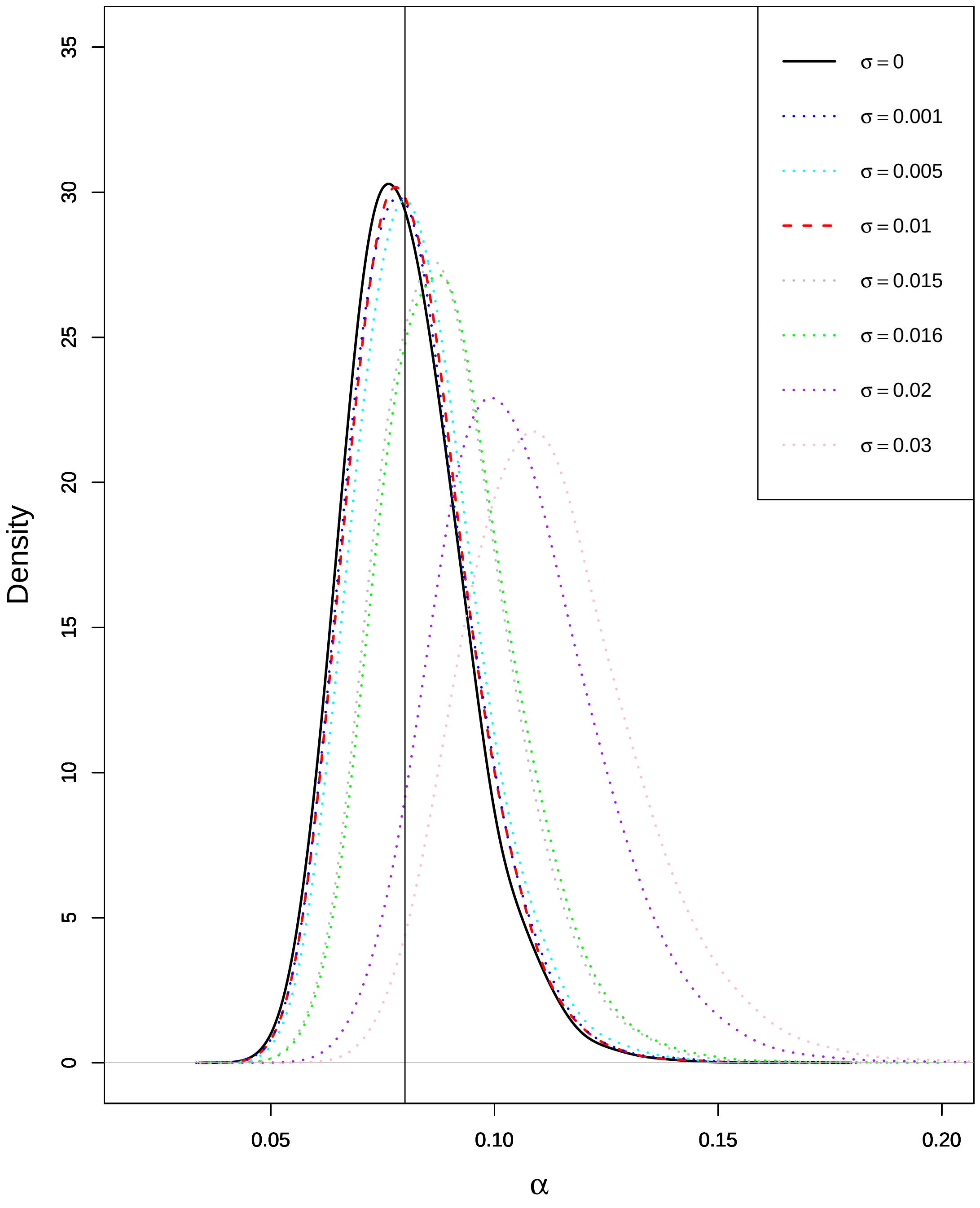

Figure 7.3.1: Posterior marginal distributions of $\alpha|y$, for data-sets in which the coordinates were perturbed after data generation with a perturbation with distribution $N(0, \sigma^2)$. The different lines plot the marginal posterior distribution density estimates for the data sets perturbed by a perturbation with standard deviation σ equal to 0, 0.001, 0.005, 0.01, 0.015, 0.016, 0.02, 0.03 km.

the spread of foot-and-mouth disease, these animal movements need to be taken into account. A possibility for further work to be performed is to discern whether there is an effect from the movement of animals. In addition, they may be an illegal movement of animals, or areas where there is an influx of illegal animals.

There may be geographical reasons to the lack of fit: the data-set contains all parcels of land within the British Isles (see fig. 7.2.8), which has a complex geography, and it may be that FMD spreads in a complex manner over distance, as this distance may include various geographical barriers for example, hills, rivers, seas.

It is interesting to compare and contrast these findings with the findings of the paper [150], in which the authors fit the same model to the same data, the FMD 2001 dataset. In the paper, the authors use forward projection to assess the adequacy of control measures used against the disease. To do so, they obtain the posterior distributions for the parameters of the epidemic, having only observed data up to a certain time, and project forward to evaluate the effect of each control measure upon the epidemic. They compared the rankings of the control measures for partially observed data up to a certain time with the observed data up to the end of the epidemic. They conclude that, after five weeks of data, the rankings of the control strategies using data collected only up to that time stabilised towards that of the rankings of the control strategies using all observed data until the end of the epidemic. As a result, in the discussion they conclude that policy recommendations from data obtained during the early stages of an epidemic outbreak can be "correct". However, this is under the assumption that the model used to obtain the control strategy rankings using all available observed data up to the end of the epidemic can give adequately accurate predictions about the trajectory of the epidemic. Increased observed amounts of data may lead to stabilised rankings of control strategies but this does not validate the rankings obtained as there is no attempt to evaluate the predictive ability of the model. This shows the importance of model criticism – examining the amount of discrepancy found between the data and the model predictions.

With regards to the results of the DILR test on the foot-and-mouth data set of 2001, with both the Cauchy and exponential kernel, the DILR test detected evidence of misfit with each of these kernels, with the DILR which was obtained with each MCMC iteration appearing to show a pattern. If the DILR was sorted by the angle between the infection link and the vector $(1,1)^T$, then the DILR appeared to show a periodic pattern. If the DILR was sorted by the cosine of the angle between the infection link and the vector $(1,1)^T$, there was a band in which there was a high concentration of directional infection link residuals which would not be expected under the null hypothesis of the DILR being uniformly distributed. It is difficult to draw conclusions from these patterns, as the bands of high concentrations of residuals do not correspond to individual angles.

It can be concluded, however, that there appears to be a phenomenon which is angle based which cannot be explained by the model for secondary infections only. The most straightforward explanation is that there is anisotropy in the spread of the foot-and-mouth disease. This may be due to unmodelled movements of animals or vehicles which have been infected or contaminated with foot-and-mouth disease. Wind could be a possible cause of anisotropy in the spread of the pathogen. Contamination of pathogens may be spread by streams or rivers. However, since neither the Cauchy nor the exponential kernel were found to be adequate using the ILR test, it is important not to put too much weight upon the conclusions of the DILR test as the apparent anisotropy may be due to inappropriate selection of the spatial kernel.

Inappropriate selection of the spatial kernel would lead to the imputation of incorrect infection links during the imputation phase of the DILR, leading to distorted values of the DILR, which could potentially lead to apparent detected anisotropy. Another possible cause for this apparent anisotropy which has been detected could be due to data quality.

As stated earlier, the data contain locations of each farm which is recorded as the location of each farmhouse, not the centroid of the parcel of land which constitutes each farm. This is in effect a perturbation upon the actual coordinates of the centroid of each parcel of land, which has been shown to affect the posterior distributions for the parameters in the model. Since the posterior distributions for the parameters are used to impute infection links in the DILR test, and since perturbation to the actual coordinates leads to inflated perception of the tail length of the spatial kernel, this may lead to infections being attributed to infectors further away than in reality. The distortion of SI links may lead to the distortion of the DILR, leading to the apparent anisotropy.

It is quite possible that there was an anisotropic perturbation: a possible cause is that if a map is being used to determine the coordinates of the farmhouse, the estimates of the northern and easting of each farmhouse could be rounded up to the nearest grid coordinate, leading to anisotropic perturbation in the X and Y axis. To determine whether this was the case, a plot of all the possible links from farm to farm against the vector $(1, 1)^T$ were plotted (see figure 7.2.7). As can be seen in this figure, the does not appear to be a "clumping" of links at any angle, which might be seen if an anisotropic perturbation with the be applied to the coordinates. Hence, the possibility of an anisotropic perturbation affecting the results is unlikely. This also rules out the possibility of there being possible links naturally clustered around certain angles, due to geography etc.

In conclusion, the analyses in this chapter appear detect misfit in the model that was fitted to the foot-and-mouth disease data of 2001. There are many possible causes of misfit, and it is not known to what extent each of these potential causes are contributing to the detected misfit. Further investigation is required on this topic.

Chapter 8

Conclusion

In this thesis, we have devised methods to assess model fit for stochastic compartmental spatio-temporal epidemic models. Many of the challenges in spatio-temporal modelling of epidemics arise from the fact that the spatial kernel, important in determining the control strategy for such epidemics, models a process that is unobserved (Chapters 1 and 2), and integrating out this unobserved data is often not analytically possible. Because of this, one of the most common ways of fitting models is through data augmented Markov chain Monte Carlo [65, 143, 64, 27, 172, 171] (Chapter 2). In this method, the missing data is treated as a nuisance parameter.

This unobserved data creates problems in devising methods for assessing model fit (Chapter 3). Bayes factors are an intuitive method of model assessment, but suffer from the drawback that it is difficult to devise algorithms to calculate the Bayes factor [91], and also suffer from other unintuitive drawbacks such as Lindley's paradox and the fact that the Bayes factor is often more sensitive to the parameter priors then the posterior distribution is sensitive to the parameter priors. Other purely Bayesian model assessment methods (for example [47]) suffer from similar drawbacks. In addition, in using Bayes factors, one must specify an alternative model or set of models to compare against. Deviance information criterion (DIC) [164] methods use the idea of parsimony to determine the best model out of several competing models. Unfortunately, there are many possible ways to deal with the unobserved data which lead to different rankings of models

[34]. It is unclear which of these model rankings should be used. The DIC for each fitted model is calculated conditioning on the fitted model, so there is no unified "viewpoint" from which to compare models. In addition, one must not lose sight of the original goal, which is to determine whether the model was fit for its purposes (for example, determining whether a control strategy would be adequate), not whether a model is parsimonious. The third set of approaches are posterior predictive checking approaches [78, 157, 158], which use a discrepancy measure to quantify discrepancy [125, 60]. Such methods can be easily incorporated into the data augmented Markov chain Monte Carlo algorithms used to fit spatio-temporal epidemic models, and do not require an alternative model or set of models to compare against. Thus, model assessment is possible with this approach, as well as model comparison. Posterior predictive checking approaches range from simple visual checks to posterior predictive tests, which can embed frequentist tests within the Bayesian framework. More complicated, multidimensional test statistics can be devised for posterior predictive testing which can target important components of models, in order to assess their fit (Chapter 3).

This thesis has presented an approach to testing that uses multidimensional test statistics, in which the idea of functional-models has been used to create test statistics which can determine whether substantial anisotropy is present (Chapter 3). This test will allow modellers to test the common assumption in models that the spatial kernel is isotropic, and allow them to determine whether there is some phenomena which causes the epidemic to spread in an anisotropic manner. Runs with simulated data showed that the proposed DILR test can detect model mis-specification, where the kernel has been specified in the fitted model as isotropic, but there is anisotropy present in the infection process (Chapter 6). This demonstrates the power of the functional-model approach in creating generalised residuals for epidemic models, which allows the creation of test statistics which target important aspects of model fit, whilst minimising the use of information from the unobserved data imputed during the model fitting stage in the model assessment stage. The discrepancy measures of anisotropy put forward in this thesis are constructed to avoid the reinforcement inherent in posterior predictive checking methods, where if too much information is reused from the model fitting stage in the model checking stage, the test will be unable to determine mis-specification.

Other test statistics can be created by embedding other tests within the posterior predictive testing framework (Chapter 3). In this thesis, the generalised likelihood ratio test was embedded within the posterior predictive testing framework, creating latent likelihood ratio tests for spatial stochastic epidemic models (Chapter 3). These methods require an alternative model to be specified and, therefore, can be viewed as model comparison techniques, as well as being ways to assess model fit. However, these requirements in specifying an alternative model allows these tests to detect mis-specification with greater ease from the alternative model in some cases. This especially seems to be the case where competing models are very similar to each other, where the full latent likelihood ratio test seems to outperform the infection link residuals test (Chapter 4). A partial LLR test was developed to determine if the test could be strengthened further (Chapter 3). Since the ILR test used information about infection order, it was thought that perhaps only the use of the infection order and not the actual infection times in a partial latent likelihood ratio test would reduce the level of reinforcement and therefore strengthen the tests. In most cases, the full LLR test still outperformed the partial LLR (Chapter 4), indicating that there may be information in the infection times that is not in the infection order that is useful in determining discrepancy between the fitted model and the actual model.

Such embedded tests are calculated at regular intervals during the data augmented MCMC algorithm and thus can be embedded in the model fitting process. An advantage of this is that MCMC iterations can be saved from the model fitting process, and processed later off-line (demonstrated in Chapter 7).

Since such tests are performed at regular intervals, and rely on nested Monte Carlo to obtain a p-value estimate when models are non-nested, these estimates require a lot of MCMC iterations to obtain high accuracy. In this thesis, we have devised GPU friendly algorithms which allow epidemic modellers to parallelise the RJMCMC and posterior predictive checks put forward in this thesis (Chapter 5). These methods will allow epidemic modellers who may typically face time pressure when performing analyses during an ongoing epidemic to have faster turnaround times. Before the year 2004, such computational techniques would not have been necessary as clock speeds of processors have risen rapidly over the previous decade allowing the same algorithms to run on a new computer a lot faster with no change to coding or computational techniques [177, 160]. Unfortunately, clock speeds have now plateaued, but instead the number of parallel processing cores on each processor has risen dramatically. This necessitates the development of the parallel computational techniques for MCMC. The structure of the GPU is built for the intensive mathematical calculations needed for the rendering of 3D graphics, and thus, such a computational processor is uniquely suited for numerical applications such as scientific, numeric and statistical programming, and the GPU algorithms in this thesis will give researchers the ability to access high-performance computation on consumer-grade desktop computers (Chapter 5). The algorithms detailed in this thesis use parallel programming patterns, and therefore can use the highly optimised code of parallel libraries with optimised routines for these parallel programming patterns to obtain portable performance (Chapter 5).

This research opens up opportunities for further study. Whilst being powerful and practical, latent likelihood ratio tests require the use of numerical algorithms to optimise the likelihood. The main drawback with this method is that it is difficult to prove that the maximum obtained in each of the several thousand iterations of the embedded test is indeed a global maximum and not a local maximum. It would be of great utility to be able to derive analytical expressions for the maximum-likelihood estimates, as this will make it certain the maximum obtained is the MLE. Such analytical methods will also lower the significant computational load of such tests.

The calculation of the latent p-value in the embedded latent likelihood ratio test depends upon Monte Carlo simulation for the case where the models that are to be compared are non-nested. When comparing different spatial kernels, this situation can be encountered frequently. An useful development would be to have an analytical expression for the distribution of the test statistic under the null hypothesis, or at least a computational method which is able to converge to high accuracy within relatively few iterations.

An additional development which would be of great benefit would be the derivation of analytical expressions of approximations for the power of latent tests based on posterior predictive checking. There are currently results that give an upper bound on the power of the latent likelihood tests but there are no results for lower bounds or estimates of the power, or under what circumstances can a minimal distance to this upper bound be obtained (Chapter 4).

The model assessment comparison and assessment methods presented here have been implemented within data augmented MCMC, but can be implemented within other algorithms such as particle filters. Such developments would be useful for the purposes of model comparison within real-time analysis of epidemic data of an ongoing epidemic, and further investigation of the pros and cons of doing so would be beneficial.

The case study on the FMD 2001 data-set (Chapter 7) shows that the methods shown here can be performed offline, and on massive data-sets. The investigations showed that there was substantial discrepancy between the fitted models and the data (Chapter 7), but it is unclear whether this is due to the quality of the data, the model or the approximations used in implementing the model and/or representation of the data. The results of the case study on the FMD epidemic of 2001 raises questions about the adequacy of the representation of hosts within an epidemic as points on the X-Y grid, when such points represent farms, or areas of land rather than individual hosts. To what extent can this approach be pursued until the approximation breaks down? Is there a point within the area at which the area of land is best represented, for example, the centre of mass or the centroid of the area?

One of the advantages of the latent likelihood ratio test is that as long as there is an expression for the likelihood, the test can be used over a large range of circumstances and applications. There are very many real-world scenarios in which two models need to be compared and there is missing or unobserved data involved. Even within the general idea of infection or infectiousness, the ideas of contact based infection may be extended to many other fields, for example the monitoring of trends in social media, such as viral tweets and memes, or prediction and control of computer viruses. The approach of designing partial likelihood or full likelihood-based tests, in which minimal information is reused from the fitting stage in the model testing stage, is applicable in all these fields, with the same benefits as in epidemic modelling. It is quite possible that the methods described in this thesis can be applied in fields which seem to be quite unconnected to epidemic modelling, for example, signal processing of LIDAR or radar data.

Bibliography

[1] Abelson, H., Sussman, G. J., & Sussman, J. (1996) *Structure and interpretation of computer programs*. The MIT Press.

[2] Adrakey, H. K. Unpublished Run Results.

[3] Adrakey, H. K. (2016) *Control and Surveillance of Partially Observed Stochastic Epidemics in a Bayesian Framework*. Ph.D. thesis, Heriot-Watt University. URL: `http://hdl.handle.net/10399/3290`.

[4] Adrakey, H. K., Streftaris, G., Cunniffe, N. J., Gottwald, T. R., Gilligan, C. A., & Gibson, G. J. (2017) Evidence-based controls for epidemics using spatio-temporal stochastic models in a Bayesian framework. *Journal of The Royal Society Interface*, **14**, 20170386. `doi:10.1098/rsif.2017.0386`.

[5] Amdahl, G. M. (1967) Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, 483–485. ACM.

[6] Anderson, I. (2002). Foot and Mouth Disease 2001: Lessons to be Learned Inquiry Report. URL: `http://webarchive.nationalarchives.gov.uk/20100809105008/http://archive.cabinetoffice.gov.uk/fmd/fmd_report/report/index.htm`.

[7] Anderson, T. W. & Darling, D. A. (1954) A Test of Goodness of Fit. *Journal of the American Statistical Association*, **49**, 765–769. `doi:10.2307/2281537`.

[8] Anthony, S. (2011). Rent the World's 30th-fastest, 30,472-core Supercomputer for $1,279 Per Hour. URL: `https://www.extremetech.com/comput`

ing/96829-rent-the-worlds-30th-fastest-30472-core-supercomputer-for-1279-per-hour.

[9] Ball, F., Mollison, D., & Scalia-Tomba, G. (1997) Epidemics with Two Levels of Mixing. *The Annals of Applied Probability*, **7**, 46–89. doi:10.2307/2245132.

[10] Ball, F. & Neal, P. (2002) A General Model for Stochastic SIR Epidemics with Two Levels of Mixing. *Mathematical Biosciences*, **180**, 73–102. doi:10.1016/s0025-5564(02)00125-6.

[11] Barreto, M. L. (2006) Infectious Diseases Epidemiology. *Journal of Epidemiology & Community Health*, **60**, 192–195. doi:10.1136/jech.2003.011593.

[12] Batcher (1980) Design of a Massively Parallel Processor. *IEEE Transactions on Computers*, **C-29**, 836–840. doi:10.1109/tc.1980.1675684.

[13] Bates, C. (16). When Foot-and-mouth Disease Stopped the UK in Its Tracks. BBC News Magazine - BBC News Website. URL: http://www.bbc.co.uk/news/magazine-35581830.

[14] Bayarri, M. J. & Berger, J. O. (1999) Quantifying Surprise in the Data and Model Verification. In *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, 53–82. Oxford University Press.

[15] BBC News (2011). Foot-and-mouth Outbreak of 2001. BBC News Website. URL: http://www.bbc.co.uk/news/uk-england-12483017.

[16] Beam, A. (2013). MCMC Using Parallel Computation. URL: www4.stat.ncsu.edu/~ghosh/TEACHING/st790abi/resources/AndyBeamPresentation.ppt.

[17] Berger, J. O. & Wolpert, R. L. (1988) *The Likelihood Principle (IMS Lecture Notes-monograph : Vol 6)*. Institute of Mathematical Statistics. URL: https://www.amazon.com/Likelihood-Principle-Ims-Lecture-Notes-Monograph/dp/0940600137?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chi

mbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0940600137.

[18] Bernabeu, A. (2016). What Is a Warp and How Is It Different from a Thread, Block or Wave in CUDA? URL: https://www.quora.com/What-is-a-warp-and-how-is-it-different-from-a-thread-block-or-wave-in-CUDA.

[19] Bernardo, J. M. & Smith, A. F. M. (Eds.) (1994) *Bayesian Theory*. John Wiley & Sons, Inc. doi:10.1002/9780470316870.

[20] Bernoulli, D. (1760) Reflexions Sur Les Avantages De L'Inoculation. *Mercure De France*, 173–190.

[21] Bezerra, M. A., dos Santos, Q. O., Santos, A. G., Novaes, C. G., Ferreira, S. L. C., & de Souza, V. S. (2016) Simplex Optimization: A Tutorial Approach and Recent Applications in Analytical Chemistry. *Microchemical Journal*, **124**, 45–54. doi:10.1016/j.microc.2015.07.023.

[22] Bird, R. & Wadler, P. (1988) *Introduction to Functional Programming*. Prentice Hall International (UK) Ltd, Hertfordshire, UK.

[23] Birnbaum, A. (1962) On the Foundations of Statistical Inference. *Journal of the American Statistical Association*, **57**, 269–306. doi:10.1080/01621459.1962.10480660.

[24] Box, G. E. P. (1976) Science and Statistics. *Journal of the American Statistical Association*, **71**, 791–799. doi:10.1080/01621459.1976.10480949.

[25] Box, G. E. P. (1980) Sampling and Bayes' Inference in Scientific Modelling and Robustness. *Journal of the Royal Statistical Society. Series A (General)*, **143**, 383–430. URL: http://www.jstor.org/stable/2982063.

[26] Boys, R. J. & Giles, P. R. (2007) Bayesian Inference for Stochastic Epidemic Models with Time-inhomogeneous Removal Rates. *Journal of Mathematical Biology*, **55**, 223–247. doi:10.1007/s00285-007-0081-y.

[27] Britton, T. & O'Neill, P. D. (2002) Bayesian Inference for Stochastic Epidemics in Populations with Random Social Structure. *Scandinavian Journal of Statistics*, **29**, 375–390. doi:10.1111/1467-9469.00296.

[28] Brown, G. D. (2015) *Application Of Heterogeneous Computing Techniques To Compartmental Spatiotemporal Epidemic Models*. Ph.D. thesis, University of Iowa. URL: https://ir.uiowa.edu/cgi/viewcontent.cgi?article=5606&context=etd.

[29] Brown, P. E., Chimard, F., Remorov, A., Rosenthal, J. S., & Wang, X. (2013) Statistical Inference and Computational Efficiency for Spatial Infectious Disease Models with Plantation Data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **63**, 467–482. doi:10.1111/rssc.12036.

[30] Campione, M. & Walrath, K. (1997). The Java Tutorial. URL: http://journals.ecs.soton.ac.uk/java/tutorial/java/threads/definition.html.

[31] Carlin, B. P. & Chib, S. (1995) Bayesian Model Choice via Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, **57**, 473–484. URL: http://www.jstor.org/stable/2346151.

[32] Cauchemez, S., Carrat, F., Viboud, C., Valleron, A. J., & Boëlle, P. Y. (2004) A Bayesian MCMC Approach to Study Transmission of Influenza: Application to Household Longitudinal Data. *Statistics in Medicine*, **23**, 3469–3487. doi:10.1002/sim.1912.

[33] Cauchy, A. L. (1829) Mémoire Sur Divers Points D'analyse. *Mémoires de l'Académie de France*, **8**, 130–138.

[34] Celeux, G., Forbes, F., Robert, C. P., & Titterington, D. M. (2006) Deviance information criteria for missing data models. *Bayesian Analysis*, **1**, 651–673. doi:10.1214/06-ba122.

[35] Centre for Agriculture and Biosciences International (2018). Invasive Species Compendium. URL: https://www.cabi.org/isc/datasheet/56921.

[36] Chis Ster, I., Singh, B. K., & Ferguson, N. M. (2009) Epidemiological Inference for Partially Observed Epidemics: The Example of the 2001 Foot and Mouth Epidemic in Great Britain. *Epidemics*, **1**, 21–34. doi:10.1016/j.epidem.2008.09.001.

[37] Connolly, J. (2015) *The Politics and Crisis Management of Animal Health Security*. Routledge. URL: https://www.amazon.com/Politics-Crisis-Management-Animal-Security/dp/1472437748?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1472437748.

[38] Cook, A., Gibson, G., Gottwald, T., & Gilligan, C. (2008) Constructing the Effect of Alternative Intervention Strategies on Historic Epidemics. *Journal of The Royal Society Interface*, **5**, 1203–1213. doi:10.1098/rsif.2008.0030.

[39] Cox, D. R. & Snell, E. J. (1968) A General Definition of Residuals. *Journal of the Royal Statistical Society. Series B (Methodological)*, 248–275. URL: http://www.jstor.org/stable/2984505.

[40] Cunniffe, N. J., Cobb, R. C., Meentemeyer, R. K., Rizzo, D. M., & Gilligan, C. A. (2016) Modeling When, Where, and How to Manage a Forest Epidemic, Motivated by Sudden Oak Death in California. *Proceedings of the National Academy of Sciences*, **113**, 5640–5645. doi:10.1073/pnas.1602153113.

[41] Dawid, A. P. & Stone, M. (1982) The Functional-Model Basis of Fiducial Inference. *The Annals of Statistics*, **10**, 1054–1067. doi:10.1214/aos/1176345970.

[42] Debye, P. (1909) Näherungsformeln Für Die Zylinderfunktionen Für Große Werte Des Arguments Und Unbeschränkt Veränderliche Werte Des Index. *Mathematische Annalen*, **67**, 535–558. doi:10.1007/bf01450097.

[43] Deeth, L. E., Deardon, R., & Gillis, D. J. (2015) Model Choice Using the Deviance Information Criterion for Latent Conditional Individual-

Level Models of Infectious Disease Spread. *Epidemiologic Methods*, **4**. doi:10.1515/em-2014-0001.

[44] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, **39**, 1–38. URL: http://babbage.cs.missouri.edu/~chengji/mlbioinfo/dempster_em.pdf.

[45] Dey, D. K. & Rao, C. R. (2005) *Bayesian Thinking, Modeling and Computation*, volume 25. Elsevier.

[46] Diggle, P. J. (2006) Spatio-temporal Point Processes, Partial Likelihood, Foot and Mouth Disease. *Statistical Methods in Medical Research*, **15**, 325–336. doi:10.1191/0962280206sm454oa.

[47] Draper, D. (1995) Assessment and Propagation of Model Uncertainty. *Journal of the Royal Statistical Society. Series B (Methodological)*, **57**, 45–97. URL: http://www.jstor.org/stable/2346087.

[48] Ferguson, N. M. (2001) The Foot-and-Mouth Epidemic in Great Britain: Pattern of Spread and Impact of Interventions. *Science*, **292**, 1155–1160. doi:10.1126/science.1061020.

[49] Ferguson, N. M., Donnelly, C. A., & Anderson, R. M. (2001) Transmission intensity and impact of control policies on the foot and mouth epidemic in Great Britain. *Nature*, **413**, 542–548. doi:10.1038/35097116.

[50] Fernando, R. (Ed.) (2005) *GPU Gems 2*. Addison-Wesley Professional.

[51] Fitt, B. D. L., McCartney, H. A., & West, J. S. (2006) Dispersal of Foliar Plant Pathogens: Mechanisms, Gradients and Spatial Patterns. In *The Epidemiology of Plant Diseases*, edited by Cooke, B. M., Jones, D. G., & Kaye, B., 159–192. Springer Netherlands, Dordrecht. doi:10.1007/1-4020-4581-6_6.

[52] Forrester, M., Pettitt, A., & Gibson, G. (2006) Bayesian inference of hospital-acquired infectious diseases and control measures given imperfect surveillance data. *Biostatistics*, **8**, 383–401. doi:10.1093/biostatistics/kxl017.

[53] Forrester, M. & Pettitt, A. N. (2005) Use of Stochastic Epidemic Modeling to Quantify Transmission Rates of Colonization With Methicillin-Resistant Staphylococcus Aureus in an Intensive Care Unit. *Infection Control & Hospital Epidemiology*, **26**, 598–606. doi:10.1086/502588.

[54] Fung, L. W. (1977) A Massively Parallel Processing Computer. In *High speed computer and algorithm organization : proceedings of the Symposium on High Speed Computer and Algorithm Organization, held at the University of Illinois April 13-15, 1977*, edited by Kuck, D. J., Lawrie, D. H., & Sameh, A. H. Academic Press.

[55] Gail, M. H. & Benichou, J. (2000) *Encyclopedia of Epidemiologic Methods*. John Wiley & Sons Inc. URL: https://www.ebook.de/de/product/3601074/mitchell_h_gail_jacques_benichou_encyclopedia_of_epidemiologic_methods.html.

[56] Gelfand, A. E. & Smith, A. F. M. (1990) Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, **85**, 398–409. doi:10.1080/01621459.1990.10476213.

[57] Gelman, A. (2013) Understanding posterior p-values. arXiv:http://www.stat.columbia.edu/~gelman/research/unpublished/ppc_understand2.pdf.

[58] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013) *Bayesian Data Analysis*. CRC press.

[59] Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004) *Bayesian Data Analysis*. Chapman & Hall/CRC, Boca Raton, Fla, 2nd ed. edition.

[60] Gelman, A., Meng, X.-L., & Stern, H. (1996) Posterior Predictive Assessment of Model Fitness Via Realized Discrepancies. *Statistica Sinica*, **6**, 733–760. URL: `http://www.jstor.org/stable/24306036`.

[61] Gelman, A. et al. (2005) Comment: Fuzzy and Bayesian p -Values and u -Values. *Statistical Science*, **20**, 380–381. `doi:10.1214/088342305000000368`.

[62] George Casella, C. R. (2005) *Monte Carlo Statistical Methods*. Springer New York. URL: `https://www.ebook.de/de/product/3007808/george_casella_christian_robert_monte_carlo_statistical_methods.html`.

[63] Gibson, G. & Gilligan, C. A. (2015) Inference and Prediction with Individual-based Stochastic Models of Epidemics. In *Biosecurity Surveillance: Quantitative Approaches*, 6, chapter 14, 253–265. CABI. `doi:10.1079/9781780643595.0253`.

[64] Gibson, G. & Renshaw, E. (1998) Estimating Parameters in Stochastic Compartmental Models Using Markov Chain Methods. *IMA Journal of Mathematics Applied in Medicine and Biology*, **15**, 19–40. `doi:10.1093/imammb/15.1.19`.

[65] Gibson, G. J. (1997) Markov Chain Monte Carlo Methods for Fitting Spatiotemporal Stochastic Models in Plant Epidemiology. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **46**, 215–233. `doi:10.1111/1467-9876.00061`.

[66] Gibson, G. J., Otten, W., Filipe, J. A. N., Cook, A., Marion, G., & Gilligan, C. A. (2006) Bayesian Estimation for Percolation Models of Disease Spread in Plant Populations. *Statistics and Computing*, **16**, 391–402. `doi:10.1007/s11222-006-0019-z`.

[67] Gibson, G. J. & Renshaw, E. (2001) Likelihood Estimation for Stochastic Compartmental Models Using Markov Chain Methods. *Statistics and Computing*, **11**, 347–358. `doi:https://doi.org/10.1023/A:1011973120681`.

[68] Gibson, G. J. & Streftaris, G. (2011) Testing and Comparing Epidemic Models Using Hybrid Approaches.

[69] Gibson, G. J., Streftaris, G., & Thong, D. (2018) Comparison and Assessment of Epidemic Models. *Statistical Science*, **33**, 19–33. `doi:10.1214/17-sts615`.

[70] Gilks, W., Richardson, S., & Spiegelhalter, D. (1996) MCMC in Practice. *New York: Chapman and Hall*.

[71] Gilligan, C. A. & van den Bosch, F. (2008) Epidemiological Models for Invasion and Persistence of Pathogens. *Annual Review of Phytopathology*, **46**, 385–418. `doi:10.1146/annurev.phyto.45.062806.094357`.

[72] Gottwald, T. R., Graham, J. H., & Schubert, T. S. (2002) Citrus Canker: The Pathogen and Its Impact. *Plant Health Progress*, **3**, 15. `doi:10.1094/php-2002-0812-01-rv`.

[73] Gottwald, T. R., Sun, X., Riley, T., Graham, J. H., Ferrandino, F., & Taylor, E. L. (2002) Geo-Referenced Spatiotemporal Analysis of the Urban Citrus Canker Epidemic in Florida. *Phytopathology*, **92**, 361–377. `doi:10.1094/phyto.2002.92.4.361`.

[74] Gray, A. (2014). Introduction to GPU Programming. Lecture materials from EPCC Course given at Heriot-Watt. URL: `http://www2.epcc.ed.ac.uk/~alang/GPUHW/`.

[75] Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732. `doi:10.1093/biomet/82.4.711`.

[76] Green, P. J. & Hastie, D. I. (2009) Reversible jump MCMC. *Genetics*, **155**, 1391–1403.

[77] Guilhoto, L. F. (2017) Applying Markov Chains to Monte Carlo Integration. URL: `http://math.uchicago.edu/~may/REU2017/REUPapers/Guilhoto.pdf`.

[78] Guttman, I. (1967) The Use of the Concept of a Future Observation in Goodness-of-Fit Problems. *Journal of the Royal Statistical Society. Series B*

(Methodological), **29**, 83–100. URL: `http://www.jstor.org/stable/2984569`.

[79] Hammersley, J. M. & Handscomb, D. C. (1964) *Monte Carlo Methods*. Springer. `doi:10.1007/978-94-009-5819-7`.

[80] Harko, T., Lobo, F. S. N., & Mak, M. K. (2014) Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates. *Applied Mathematics and Computation*, **236**, 184–194. `doi:10.1016/j.amc.2014.03.030`.

[81] Harris, M. (2007) Optimizing parallel reduction in CUDA. **21**, 104–110.

[82] Harris, M., Owens, J. D., Sengupta, S., Tzeng, S., Zhang, Y., Davidson, A., Patel, R., Wang, L., Ashkiani, S., Mak, J., Patney, A., Yan, E., Alcantara, D., Satish, N., Ahrens, J., Aila, T., Bell, N., Buck, I., Blelloch, G., Bolz, J., Garland, M., Inman, J., Lengyel, E., Laine, S., Luebke, D., McCormick, P., Merrill, D., & Vuduc, R. THe CUDPP Library. URL: `http://cudpp.github.io`.

[83] Harris, M., Sengupta, S., & Owens, J. D. (2007) Parallel Prefix Sum (Scan) with CUDA. In *GPU Gems 3*, chapter 39, 851–876. Addison Wesley.

[84] Harris, M. J., Baxter, W. V., Scheuermann, T., & Lastra, A. (2003) Simulation of Cloud Dynamics on Graphics Hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, 92–101. Eurographics Association.

[85] Hastings, W. K. (1970) Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, **57**, 97–109. `doi:10.1093/biomet/57.1.97`.

[86] Horn, D. (2005) Stream reduction operations for GPGPU applications. In *GPU Gems*, volume 2. URL: `https://developer.nvidia.com/gpugems/GPUGems2/gpugems2_chapter36.html`.

[87] House, T., Baguelin, M., Hoek, A. J. V., White, P. J., Sadique, Z., Eames, K., Read, J. M., Hens, N., Melegaro, A., Edmunds, W. J., & Keeling, M. J. (2011) Modelling the impact of local reactive school closures on critical care provision during an influenza pandemic. *Proceedings of the Royal Society B: Biological Sciences*, **278**, 2753–2760. doi:10.1098/rspb.2010.2688.

[88] Intel (2012). Chip Shot: Intel Names the Technology to Revolutionize the Future of HPC - Intel Xeon Phi Product Family. URL: https://newsroom.intel.com/chip-shots/chip-shot-intel-names-the-technology-to-revolutionize-the-future-of-hpc-intel-xeon-phi-product-family/.

[89] Jacob, R. & Anderson, J. (1992) Do-it-Yourself Massively Parallel Supercomputer does useful Physics. *Computers in Physics*, **6**, 244. doi:10.1063/1.4823073.

[90] Jarrad, F., Low-Choy, S., & Mengersen, K. (Eds.) (2015) *Biosecurity Surveillance: Quantitative Approaches*. 6. CABI. doi:10.1079/9781780643595.0000.

[91] Jeffreys, H. (1939) *Theory of probability*. The Clarendon press.

[92] Jeffreys, H. (1961) *Theory of probability*. The Clarendon press, 3 edition.

[93] Jewell, C. (2015). InFER: Inference For Epidemic related Risk, Branch: "fmdgpu". URL: https://github.com/chrism0dwk/infer/tree/fmdgpu.

[94] Jewell, C. (2015) MCMC Samples.

[95] Jewell, C. P. & Brown, R. G. (2015) Bayesian Data Assimilation Provides Rapid Decision Support for Vector-borne Diseases. *Journal of The Royal Society Interface*, **12**, 20150367. doi:10.1098/rsif.2015.0367.

[96] Jewell, C. P., Keeling, M. J., & Roberts, G. O. (2009) Predicting Undetected Infections during the 2007 Foot-and-mouth Disease Outbreak. *Journal of The Royal Society Interface*, **6**, 1145–1151. doi:10.1098/rsif.2008.0433.

[97] Jewell, C. P., Kypraios, T., Neal, P., & Roberts, G. O. (2009) Bayesian Analysis for Emerging Infectious Diseases. *Bayesian Analysis*, **4**, 465–496. doi:10.1214/09-ba417.

[98] Johnson, S. G. The Nlopt Nonlinear-optimization Package. URL: http://ab-initio.mit.edu/nlopt.

[99] Kass, R. E. & Raftery, A. E. (1995) Bayes Factors. *Journal of the American Statistical Association*, **90**, 773–795. doi:10.1080/01621459.1995.10476572.

[100] Keeling, M. J. (2001) Dynamics of the 2001 UK Foot and Mouth Epidemic: Stochastic Dispersal in a Heterogeneous Landscape. *Science*, **294**, 813–817. doi:10.1126/science.1065973.

[101] Keeling, M. J. & White, P. J. (2010) Targeting Vaccination against Novel Infections: Risk, Age and Spatial Structure for Pandemic Influenza in Great Britain. *Journal of The Royal Society Interface*, **8**, 661–670. doi:10.1098/rsif.2010.0474.

[102] Kermack, W. O. & McKendrick, A. G. (1927) A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **115**, 700–721. doi:10.1098/rspa.1927.0118.

[103] Kirk, D. B. & Wen-mei, W. H. (2012) *Programming Massively Parallel Processors: A Hands-on Approach*. Newnes.

[104] Knock, E. S. & O'Neill, P. D. (2013) Bayesian Model Choice for Epidemic Models with Two Levels of Mixing. *Biostatistics*, **15**, 46–59. doi:10.1093/biostatistics/kxt023.

[105] Kolda, T. G., Lewis, R. M., & Torczon, V. (2003) Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, **45**, 385–482. doi:10.1137/s003614450242889.

[106] Kuiper, N. H. (1960) Tests Concerning Random Points on a Circle. In *Nederl. Akad. Wetensch. Proc. Ser. A*, volume 63, 38–47.

[107] Kypraios, T. (2007) *Efficient Bayesian inference for partially observed stochastic epidemics and a new class of semi-parametric time series models*. Ph.D. thesis. URL: `http://www.research.lancs.ac.uk/portal/services/downloadRegister/310801/kypraios.pdf`.

[108] Kypraios, T., Neal, P., & Prangle, D. (2017) A Tutorial Introduction to Bayesian Inference for Stochastic Epidemic Models Using Approximate Bayesian Computation. *Mathematical Biosciences*, **287**, 42–53. `doi:10.1016/j.mbs.2016.07.001`.

[109] Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998) Convergence Properties of the Nelder–Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization*, **9**, 112–147. `doi:10.1137/s1052623496303470`.

[110] Landau, W. (2013). CUDA C: K-means and MCMC. URL: `https://wlandau.github.io/gpu/lectures/cudac-examples/cudac-examples.pdf`.

[111] Lau, M. S. Y., Marion, G., Streftaris, G., & Gibson, G. J. (2014) New Model Diagnostics for Spatio-temporal Systems in Epidemiology and Ecology. *Journal of The Royal Society Interface*, **11**, 20131093–20131093. `arXiv:http://rsif.royalsocietypublishing.org/content/11/93/20131093.full.pdf+html`, `doi:10.1098/rsif.2013.1093`.

[112] Lindley, D. V. (1957) A Statistical Paradox. *Biometrika*, **44**, 187. `doi:10.2307/2333251`.

[113] Little, J. D. C. (1961) A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research*, **9**, 383–387. `doi:10.1287/opre.9.3.383`.

[114] Liu, D. C. & Nocedal, J. (1989) On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical Programming*, **45**, 503–528. `doi:10.1007/bf01589116`.

[115] Luebke, D., Owens, J., & Connors, D. (2013). Udacity CS344: Intro to Parallel Programming. URL: `https://developer.nvidia.com/udacity-cs344-intro-parallel-programming`.

[116] Marsaglia, G. & Marsaglia, J. (2004) Evaluating the Anderson-Darling Distribution. *Journal of Statistical Software*, **9**. `doi:10.18637/jss.v009.i02`.

[117] Martin, P. J., Ayuso, L. F., Torres, R., & Gavilanes, A. (2012) Algorithmic Strategies for Optimizing the Parallel Reduction Primitive in CUDA. In *2012 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. `doi:10.1109/hpcsim.2012.6266966`.

[118] Martin, R. P., Vahdat, A. M., Culler, D. E., & Anderson, T. E. (1997) Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture. *ACM SIGARCH Computer Architecture News*, **25**, 85–97. `doi:10.1145/384286.264146`.

[119] McCarthy, J. (1960) Recursive Functions Symbolic Expressions and Their Computation by Machine, Part I. *Communications of the ACM*, **3**, 184–195. `doi:10.1145/367177.367199`.

[120] McCool, M. D. (2010) Structured Parallel Programming with Deterministic Patterns. In *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, 5–5. USENIX Association.

[121] McKendrick, A. G. (1925) Applications of Mathematics to Medical Problems. *Proceedings of the Edinburgh Mathematical Society*, **44**, 98. `doi:10.1017/s0013091500034428`.

[122] McKinnon, K. I. M. (1998) Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization*, **9**, 148–158. `doi:10.1137/s1052623496303482`.

[123] Meentemeyer, R. K., Cunniffe, N. J., Cook, A. R., Filipe, J. A. N., Hunter, R. D., Rizzo, D. M., & Gilligan, C. A. (2011) Epidemiological modeling

of invasion in heterogeneous landscapes: spread of sudden oak death in California (1990–2030). *Ecosphere*, **2**, art17. doi:10.1890/es10-00192.1.

[124] Meijer, E., Fokkinga, M., & Paterson, R. (1991) Functional programming with bananas, lenses, envelopes and barbed wire. In *Conference on Functional Programming Languages and Computer Architecture*, 124–144. Springer, Springer Berlin Heidelberg. doi:10.1007/3540543961_7.

[125] Meng, X.-L. (1994) Posterior Predictive P-values. *The Annals of Statistics*, **22**, 1142–1160. doi:10.1214/aos/1176325622.

[126] Meng, X.-L. & Vaida, F. (2006) Comment on article by Celeux et al. *Bayesian Analysis*, **1**, 687–698.

[127] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953) Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, **21**, 1087–1092. doi:10.1063/1.1699114.

[128] MONTO, A. S., KOOPMAN, J. S., & LONGINI, I. M. (1985) Tecumseh Study of Illness XIII. Influenza Infection and Disease, 1976–1981. *American Journal of Epidemiology*, **121**, 811–822. doi:10.1093/oxfordjournals.aje.a114052.

[129] Morris, R. S., Stern, M. W., Stevenson, M. A., Wilesmith, J. W., & Sanson, R. L. (2001) Predictive Spatial Modelling of Alternative Control Strategies for the Foot-and-mouth Disease Epidemic in Great Britain, 2001. *Veterinary Record*, **149**, 137–144. doi:10.1136/vr.149.5.137.

[130] Neal, P. & Huang, C. L. T. (2014) Forward Simulation Markov Chain Monte Carlo with Applications to Stochastic Epidemic Models. *Scandinavian Journal of Statistics*, **42**, 378–396. doi:10.1111/sjos.12111.

[131] Neal, P. J. & Roberts, G. O. (2004) Statistical Inference and Model Selection for the 1861 Hagelloch Measles Epidemic. *Biostatistics*, **5**, 249–261. doi:10.1093/biostatistics/5.2.249.

[132] Nelder, J. A. & Mead, R. (1965) A Simplex Method for Function Minimization. *The Computer Journal*, **7**, 308–313. doi:10.1093/comjnl/7.4.308.

[133] Nemire, B. & Jewell, C. (2015). Using GPUs to Accelerate Epidemic Forecasting. URL: https://devblogs.nvidia.com/gpus-accelerate-epidemic-forecasting/.

[134] Neri, F. M., Cook, A. R., Gibson, G. J., Gottwald, T. R., & Gilligan, C. A. (2014) Bayesian Analysis for Inference of an Emerging Epidemic: Citrus Canker in Urban Landscapes. *PLoS Computational Biology*, **10**, e1003587. doi:10.1371/journal.pcbi.1003587.

[135] Nickolls, J., Buck, I., Garland, M., & Skadron, K. (2008) Scalable Parallel Programming with CUDA. *Queue*, **6**, 40. doi:10.1145/1365490.1365500.

[136] Nickolls, J. & Dally, W. J. (2010) The GPU Computing Era. *IEEE Micro*, **30**, 56–69. doi:10.1109/MM.2010.41.

[137] Nocedal, J. (1980) Updating Quasi-newton Matrices with Limited Storage. *Mathematics of Computation*, **35**, 773–773. doi:10.1090/s0025-5718-1980-0572855-7.

[138] NVIDIA (2014) *CUDA C Programming Guide*. NVIDIA.

[139] NVIDIA Corporation (2018). Cuda FAQ. URL: https://developer.nvidia.com/cuda-faq.

[140] O'Neill, P. D. (2001) Inference for an Epidemic When Susceptibility Varies. *Biostatistics*, **2**, 99–108. doi:10.1093/biostatistics/2.1.99.

[141] O'Neill, P. D., Balding, D. J., Becker, N. G., Eerola, M., & Mollison, D. (2000) Analyses of Infectious Disease Data from Household Outbreaks by Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **49**, 517–542. doi:10.1111/1467-9876.00210.

[142] O'Neill, P. D. & Kypraios, T. (2014) Bayesian model choice via mixture distributions with application to epidemics and population process models. ArXiv:1411.7888 [stat.ME]. URL: `http://arxiv.org/abs/1411.7888`.

[143] O'Neill, P. D. & Roberts, G. O. (1999) Bayesian Inference for Partially Observed Stochastic Epidemics. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **162**, 121–129. `doi:10.1111/1467-985x.00125`.

[144] Oxford University Press (2018). Oxford Living Dictionaries - British and World English. URL: `https://en.oxforddictionaries.com/definition/api`.

[145] Parry, M., Gibson, G. J., Parnell, S., Gottwald, T. R., Irey, M. S., Gast, T. C., & Gilligan, C. A. (2014) Bayesian Inference for an Emerging Arboreal Epidemic in the Presence of Control. *Proceedings of the National Academy of Sciences*, **111**, 6258–6262. `arXiv:http://www.pnas.org/content/111/17/6258.full.pdf+html`, `doi:10.1073/pnas.1310997111`.

[146] PC.net (2018). PC.net Glossary. URL: `https://pc.net/glossary/`.

[147] Porta, M. (Ed.) (2014) *A Dictionary of Epidemiology*. Oxford University Press, 6th edition. `doi:10.1093/acref/9780195314496.001.0001`.

[148] Preshing, J. (2013). Atomic vs. Non-Atomic Operations. URL: `http://preshing.com/20130618/atomic-vs-non-atomic-operations/`.

[149] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007) *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

[150] Probert, W. J. M., Jewell, C. P., Werkman, M., Fonnesbeck, C. J., Goto, Y., Runge, M. C., Sekiguchi, S., Shea, K., Keeling, M. J., Ferrari, M. J., & Tildesley, M. J. (2018) Real-time Decision-making during Emergency Disease Outbreaks. *PLOS Computational Biology*, **14**, e1006202. `doi:10.1371/journal.pcbi.1006202`.

[151] Rao, A. S. R. S. & Maini, P. K. (2016) Influencing HIV/AIDS Policy in India Through Mathematical Modelling. In *UK Success Stories in Industrial Mathematics*, 257–261. Springer International Publishing. doi:10.1007/978-3-319-25454-8_33.

[152] Rieux, A., Soubeyrand, S., Bonnot, F., Klein, E. K., Ngando, J. E., Mehl, A., Ravigne, V., Carlier, J., & de Lapeyre de Bellaire, L. (2014) Long-Distance Wind-Dispersal of Spores in a Fungal Plant Pathogen: Estimation of Anisotropic Dispersal Kernels from an Extensive Field Experiment. *PLoS ONE*, **9**, e103225. doi:10.1371/journal.pone.0103225.

[153] Robert, C. P. (2014) On the Jeffreys-Lindley Paradox. *Philosophy of Science*, **81**, 216–232. doi:10.1086/675729.

[154] Robert, C. P. (2015) The Metropolis-Hastings algorithm. arXiv:http://arxiv.org/abs/1504.01896v3, doi:10.1002/9781118445112.stat07834.

[155] Rowan, T. H. (1990) *Functional Stability Analysis of Numerical Algorithms*. Ph.D. thesis, University of Texas at Austin, University of Texas at Austin.

[156] Rowlingson, B. (2015). InFER: Inference For Epidemic-related Risk GSOC 2015 Project. URL: https://github.com/rstats-gsoc/gsoc2015/wiki/InFER.

[157] Rubin, D. B. (1981) Estimation in Parallel Randomized Experiments. *Journal of Educational Statistics*, **6**, 377. doi:10.2307/1164617.

[158] Rubin, D. B. (1984) Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician. *The Annals of Statistics*, **12**, 1151–1172. doi:10.1214/aos/1176346785.

[159] Rubinstein, R. Y. (1981) *Simulation and the Monte Carlo Method (Wiley Series in Probability and Statistics)*. Wiley-Interscience. URL: https://www.amazon.com/Simulation-Monte-Method-Probability-Statistics/dp/0471089176?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0471089176.

[160] Scott, S. (2012). No Free Lunch for Intel MIC (or GPUs). URL: `https://blogs.nvidia.com/blog/2012/04/03/no-free-lunch-for-intel-mic-or-gpus/`.

[161] Sellke, T. (1983) On the asymptotic distribution of the size of a stochastic epidemic. *Journal of Applied Probability*, **20**, 390–394. `doi:10.2307/3213811`.

[162] Sengupta, S., Harris, M., Garland, M., & Owens, J. D. (2011) Efficient Parallel Scan Algorithms for many-core GPUs. In *Scientific Computing with Multicore and Accelerators*, edited by Kurzak, J., Bader, D. A., & Dongarra, J., chapter 19, 413–442. Chapman & Hall/CRC Computational Science.

[163] Sengupta, S., Harris, M., Zhang, Y., & Owens, J. D. (2007) Scan primitives for GPU computing. In *Graphics hardware*, volume 2007, 97–106.

[164] Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002) Bayesian Measures of Model Complexity and Fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **64**, 583–639. `doi:10.1111/1467-9868.00353`.

[165] Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2014) The Deviance Information Criterion: 12 Years On. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **76**, 485–493. `doi:10.1111/rssb.12062`.

[166] Starr, J., Campbell, A., Renshaw, E., Poxton, I., & Gibson, G. (2009) Spatio-temporal Stochastic Modelling of Clostridium Difficile. *Journal of Hospital Infection*, **71**, 49–56. `doi:10.1016/j.jhin.2008.09.013`.

[167] Steele, G. L. (1982) An overview of COMMON LISP. In *Proceedings of the 1982 ACM symposium on LISP and functional programming - LFP '82*. ACM Press. `doi:10.1145/800068.802140`.

[168] Ster, I. C. & Ferguson, N. M. (2007) Transmission Parameters of the 2001 Foot and Mouth Epidemic in Great Britain. *PLoS ONE*, **2**, e502. `doi:10.1371/journal.pone.0000502`.

[169] Stockdale, J. E., Kypraios, T., & O'Neill, P. D. (2017) Modelling and Bayesian Analysis of the Abakaliki Smallpox Data. *Epidemics*, **19**, 13–23. `doi:10.1016/j.epidem.2016.11.005`.

[170] Stratton, J. A., Anssari, N., Rodrigues, C., Sung, I.-J., Obeid, N., Chang, L., Liu, G. D., & Hwu, W.-m. (2012) Optimization and Architecture Effects on GPU Computing Workload Performance. In *Innovative Parallel Computing-Foundations & Applications of GPU, Manycore, and Heterogeneous Systems (INPAR 2012)*, 1–10. IEEE.

[171] Streftaris, G. & Gibson, G. J. (2004) Bayesian Analysis of Experimental Epidemics of Foot-and-mouth Disease. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1111–1118. `doi:10.1098/rspb.2004.2715`.

[172] Streftaris, G. & Gibson, G. J. (2004) Bayesian Inference for Stochastic Epidemics in Closed Populations. *Statistical Modelling*, **4**, 63–75. `doi:10.1191/1471082x04st065oa`.

[173] Streftaris, G. & Gibson, G. J. (2012) Non-exponential Tolerance to Infection in Epidemic Systems - Modeling, Inference, and Assessment. *Biostatistics*, **13**, 580–593. `doi:10.1093/biostatistics/kxs011`.

[174] Strohmaier, E., Dongarra, J., Simon, H., & Meuer, M. (2018). Top 500 Green Supercomputers List. URL: `https://www.top500.org/green500/`.

[175] Strohmaier, E., Dongarra, J., Simon, H., & Meuer, M. (2018). Top 500 Supercomputers List. URL: `https://www.top500.org/`.

[176] Suchard, M. A., Wang, Q., Chan, C., Frelinger, J., Cron, A., & West, M. (2010) Understanding GPU Programming for Statistical Computation: Studies in Massively Parallel Massive Mixtures. *Journal of Computational and Graphical Statistics*, **19**, 419–438. `doi:10.1198/jcgs.2010.10016`.

[177] Sutter, H. (2005). The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software. URL: `http://www.gotw.ca/publications/concurrency-ddj.htm`.

[178] Tanner, M. A. & Wong, W. H. (1987) The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, **82**, 528–540. doi:10.1080/01621459.1987.10478458.

[179] Tanner, M. A. & Wong, W. H. (2010) From EM to Data Augmentation: The Emergence of MCMC Bayesian Computation in the 1980s. *Statistical Science*, **25**, 506–516. doi:10.1214/10-sts341.

[180] Tatem, A. J., Rogers, D. J., & Hay, S. I. (2006) Global Transport Networks and Infectious Disease Spread. In *Advances in Parasitology*, 293–343. Elsevier. doi:10.1016/s0065-308x(05)62009-x.

[181] Terenin, A., Dong, S., & Draper, D. (2018) GPU-accelerated Gibbs Sampling: A Case Study of the Horseshoe Probit Model. *Statistics and Computing*. doi:10.1007/s11222-018-9809-3.

[182] Tildesley, M. J., Deardon, R., Savill, N. J., Bessell, P. R., Brooks, S. P., Woolhouse, M. E., Grenfell, B. T., & Keeling, M. J. (2008) Accuracy of Models for the 2001 Foot-and-mouth Epidemic. *Proceedings of the Royal Society B: Biological Sciences*, **275**, 1459–1468. doi:10.1098/rspb.2008.0006.

[183] Tzeng, S. & Wei, L.-Y. (2008) Parallel White Noise Generation on a GPU Via Cryptographic Hash. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games - SI3D '08*. ACM Press. doi:10.1145/1342250.1342263.

[184] U.S. Department of Health & Human Services (2017). SARS Basics Fact Sheet. URL: https://www.cdc.gov/sars/about/fs-sars.html.

[185] Volkov, V. (2010) Better Performance at Lower Occupancy. In *Proceedings of the GPU technology conference, GTC 2010*, volume 10, 16. San Jose, CA. URL: https://www.nvidia.com/content/GTC-2010/pdfs/2238_GTC2010.pdf.

[186] Volkov, V. (2016) *Understanding Latency Hiding on GPUs*. Ph.D. thesis, UC Berkeley. URL: http://digitalassets.lib.berkeley.edu/etd/ucb/text/Volkov_berkeley_0028E_16465.pdf.

[187] Volkov, V. & Demmel, J. W. (2008) Benchmarking GPUs to Tune Dense Linear Algebra. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, 1–11. IEEE, IEEE. doi:10.1109/SC.2008.5214359.

[188] WHO (2018). Health Topics: Infectious Diseases. URL: https://www.who.int/topics/infectious_diseases/en/.

[189] Wood, S. N. (2015) *Core Statistics*. Institute of Mathematical Statistics Textbooks. Cambridge University Press. doi:10.1017/CBO9781107741973.

[190] Yong-Young, K. (2003). Your Next Supercomputer – Playstation 2? URL: https://www.zdnet.com/article/your-next-supercomputer-playstation-2/.