UNIVERSITY OF CALIFORNIA

Los Angeles

Bayesian Learning and Calibration of Mechanistic Models and Spatiotemporal Computer Simulations

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Biostatistics

by

Ian Daniel Frankenburg

2022

ABSTRACT OF THE DISSERTATION

Bayesian Learning and Calibration of Mechanistic Models

and Spatiotemporal Computer Simulations

by

Ian Daniel Frankenburg

Doctor of Philosophy in Biostatistics

University of California, Los Angeles, 2022

Professor Sudipto Banerjee, Chair

Melding of information from observed data, computer simulations, and scientifically-driven mechanistic models is evolving into standard practice in diverse disciplines. This dissertation presents applications and methodology for probabilistic modeling, learning, and inference in this environment. Benefits from modeling with additional scientific and domain-specific information are numerous, as methods can be tailored to answer more focused questions, test hypotheses, and make improved forecasts.

Our first application demonstrates the utility of modeling early Covid-19 dynamics in New York City with a hierarchical Bayesian structural model. Parameter calibration is achieved using multiple data streams, consisting of cell phone movement data and disease case counts over time. The parameters of the model have specific scientific importance, and this enables both improved process understanding and more accurate forecasting based on limited data. We demonstrate with out-of-sample forecasting and sensitivity analyses.

Though qualitatively and quantitatively desirable, not all scientifically-inspired models and simulations are computationally tenable for statistical inference. This is inspiration for the later chapters of the dissertation. As large scale data becomes more abundant, new methodologies and algorithms are necessary to make feasible the data fusion process of combining expensive phenomenological models and statistical or machine learning methods. This is particularly important for costly spatiotemporal simulations that arise across the physical sciences. In the spatiotemporal setting, phenomena often evolve in time and space that follow complex dynamics and require extensive computational experimentation and simulation to adequately model. These expensive spatiotemporal computer models can thus be difficult to calibrate to real-world data. This motivates our methodological development towards learning and calibrating expensive spatiotemporal computer models. We make use of state-space methodology and multiple Gaussian stochastic processes to build an efficient statistical emulator to replace the expensive computational model. The fast emulator is then used to calibrate to observed data. This model structure facilitates efficient recursive computing and sampling of parameters to provide full uncertainty quantification. We develop these inferential algorithms to make use of parallel computing and reduced rank Gaussian spatial processes for scalability to large datasets.

In our applications, we show the methodology can learn the form of complicated dynamics arising from systems of ordinary and partial nonlinear differential equations, as well as computational models with no algebraic form. This provides a black-box learning approach for the applied researcher when modeling data with expensive simulations. We hope this continues to advance research towards developing frameworks for mixing statistical, mechanistic, and computational simulations for modeling across the sciences and beyond.

The dissertation of Ian Daniel Frankenburg is approved

Roch A.K. Nianogo

Marc A. Suchard

Donatello Telesca

Sudipto Banerjee, Committee Chair

University of California, Los Angeles

2022

To Mom, Annie, Caroline, and Jack.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Sudipto Banerjee for his guidance and advice throughout my Ph.D. at UCLA. I particularly appreciate his willingness to support my ideas, curiosities, and ambitions, while inspiring maturity and confidence in my work. This has contributed greatly towards my personal growth and professional development into an independent researcher and statistician. Dr. Donatello Telesca also provided kind mentorship and exposure to broad methodology and research as an early student at UCLA, for which I am very thankful. I would lastly like to acknowledge Dr. Marc Suchard for teaching me appreciation of application, methodology, and computing within the statistical sciences.

I have been lucky enough to build life-long friendships at UCLA that have shaped who I am today. The friendships built within the Department of Biostatistics and Department of Mathematics have made this a rewarding journey, and I value these relationships even more than the knowledge gained throughout my research and studies.

Finally, I am most grateful for the love and support of my family. This would not have been possible without them.

VITA

2012–2016 B.A. Computer Science and Mathematics, The Ohio State University

2017–2022 Ph.D. Student and Candidate in Biostatistics, University of California, Los Angeles

PUBLICATIONS

Frankenburg I, Banerjee S. (2022) Dynamic Bayesian Learning and Calibration of Spatiotemporal Computer Models. (In preperation).

Frankenburg I, Banerjee S. SSGP: an R package for Bayesian Emulation, Calibration, and Uncertainty Quantification using State-space models and Gaussian Processes. (In preperation).

Frankenburg I, Banerjee S. (2021) A Compartment Model of Human Mobility and Early Covid-19 Dynamics in NYC. *The New England Journal of Statistics in Data Science*.

CHAPTER 1

Introduction

1.1 Dissertation Outline and Contributions

This dissertation proceeds from a preliminary Chapter 2, motivating the central objective of this research towards building a framework for Bayesian modeling with a diverse set of mechanistic models and computer simulations. We provide context and background derivations by synthesizing results necessary for the methodology presented in the remaining chapters of the dissertation.

In Chapter 3, we build a mechanistic system to model the relation between a reduction in human mobility and early Covid-19 spread dynamics within New York City. To this end, we propose a multivariate compartmental model that jointly models smartphone mobility data and case counts during the early phase of the pandemic in NYC. To connect to the observed data, we combine beta regression and negative binomial regression to jointly model the separate time series. Parameter calibration is achieved through the formulation of a general Bayesian hierarchical model to provide uncertainty quantification of resulting estimates. The open-source probabilistic programming language Stan is used for the requisite computation, making use of Hamiltonian Monte Carlo. Through sensitivity analysis and out-of-sample forecasting, we find our interpretable structural model provides evidence that reductions in human mobility altered case dynamics while achieving good predictive performance.

Chapter 4 develops a computationally tractable approach for Bayesian learning and calibration of spatiotemporal computer models. We combine state-space methods and multiple Gaussian processes to enable efficient recursive computation over time and joint dynamic modeling of the computer input

space and spatial domain. By exploiting model conjugacy and reduced-rank Gaussian processes, our methodology is applicable to large-scale spatiotemporal calibration problems. We incorporate a dynamic bias term to address computer model misspecification and when the emulator may fail. This builds a coherent framework for melding information from space-time field data and expensive computational experiments. We apply our method to computer model inverse problems arising in the analysis of ordinary and partial nonlinear differential equations, as well as a computational model with no algebraic form.

Chapter 5 concludes with further work and ideas towards development of an efficient emulator for nonlinear computer models. This methodology makes use of multiprocess state-space methods to enable mixing of multiple state-space Gaussian process emulators. We provide a new sampling algorithm, with implementation and methodological refinement left for future work.

1.2 Software, Model Extensions, and Future Directions

To reproduce the work in this dissertation, all software and datasets are available as open-source. The Stan implementation of the SLIR statistical-mechanistic model of Chapter 3 is available at `https://github.com/ian-frankenburg/SLIR`. Additionally, the time series datasets that include cell phone mobility and daily case counts are included as `csv` files. The spatiotemporal model of Chapter 4 is available at `https://github.com/ian-frankenburg/SSGP`, along with scripts replicating the data analyses and applications. This code will eventually be refined and released as a statistical package for the applied researcher to enable black-box emulation, calibration, and uncertainty quantification for computer models. In Chapter 5, we detail future work and extensions towards building a surrogate model capable of capturing complex nonlinear computer simulator behavior. This is accomplished through multiprocess state-space methods (West and Harrison, 1997). This model extension introduces a latent indicator variable to enable mixing of multiple state-space Gaussian process emulators. Building upon the work in Chapter 4, this will allow us to form mixtures of Gaussian processes across complicated domains. Further extensions involve inducing time-varying

dynamics to the indicator, known as a Markov-switching state-space model. As Hidden Markov models (HMMs) are a fundamental building block of this work, we release a simple Bayesian Hidden Markov model implementation as the foundation, with the extension to state-space Gaussian process emulators to be developed. This Bayesian HMM is implemented in `C++` with available `R` calls hosted at `https://github.com/ian-frankenburg/HMMbayes`.

CHAPTER 2

Preliminaries

2.1 Modeling with Mechanisms and Computer Simulations

Mechanistic models and computer simulations are useful tools for understanding qualitative and quantitative behavior of complex systems, ranging from pharmacodynamics to infectious disease spread to econometric modeling. These models are centrally motivated through an explicit mapping between the model parameters and the scientific or physical process under study (Baker et al., 2018). Early work in this dissertation models with an epidemiologically-motivated coupled set of differential equations. Typically, such expressions admit no analytic solutions, so numerical methods are required for parameter calibration and statistical inference. These numerical solutions are often fast to compute and amenable to direct incorporation into a sampling or optimization routine.

However, many mechanistic models are computationally demanding or simulation-based, especially in the spatiotemporal setting. In the physical world, spatiotemporal processes can be complex and high-dimensional, so the underlying processes might not be fully understood without large collaborative efforts to build realistic simulators. Enabling modeling and connecting to real-world data in such an environment is an open problem and motivates much of our central methodology. This methodology draws from many areas of statistical modeling, and the structure of this chapter is as follows. In the first section, we review Bayesian state-space models, a flexible dynamic class of models that admit some structural and mechanistic parameterization. We derive necessary algebraic expressions and background material for use in Chapter 3 and 4. The chapter concludes with a section developing the emulation/calibration paradigm, which enables modeling of complex systems through

a functional surrogate.

2.2 Bayesian State-Space Models

State-space methods provide a unified framework for modeling a wide range of temporal and spatiotemporal data by allowing parameters to dynamically change and adapt according to some stochastic process over time, space, or both. In this sense, state-space models may be viewed as extended ordinary regression models. These models have a long history in mathematics, science, and engineering dating back to Russian mathematician Andrey Kolmogorov and Norbert Wiener (Wiener, 1964; Kolmogorov, 1941). The methodological focus of Chapter 4 relies on a Gaussian spatiotemporal state-space model construction, so this section illustrates background material and important derivations. By way of notation, define the following.

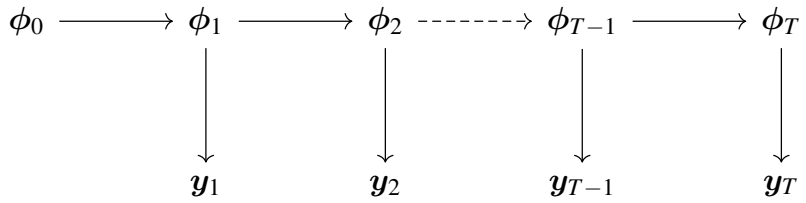
- $\boldsymbol{y}_t \in \mathbb{R}^n$ is the measurement at time t and $\boldsymbol{y}_{1:T} = (\boldsymbol{y}_1, \dots \boldsymbol{y}_T)^\top \in \mathbb{R}^{nT}$. .
- $\boldsymbol{\phi}_t \in \mathbb{R}^p$ is the latent state of the system evolving at time t and $\boldsymbol{\phi}_{0:T} = (\boldsymbol{\phi}_0, \dots \boldsymbol{\phi}_T)^\top \in \mathbb{R}^{pT}$
- $p(\boldsymbol{y}_t \mid \boldsymbol{\phi}_t)$ is the measurement model to describe the distribution of observations conditional upon the state.
- $p(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1})$ is the dynamic model which describes the stochastic dynamics of the underlying system or regression parameters. The dynamics are flexible and can be specified through a probability density, counting measure, or mixture.

Definition (Probabilistic state-space model)**.** *A probabilistic state-space model consists of a sequence of conditional probability distributions:*

$$\begin{aligned} \boldsymbol{y}_t &\sim p(\boldsymbol{y}_t \mid \boldsymbol{\phi}_t) \\ \boldsymbol{\phi}_t &\sim p(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1}) \end{aligned} \tag{2.1}$$

for $t = 1, \dots, T$.

As a directed acyclic graph (DAG), the model is represented as



This model structure is Markovian, meaning two properties are most important.

Property (Markovian state dynamics)**.** *For time $t = 1, \ldots, T$, the states $\{\phi_t\}_{t=0}^{T}$ form a Markov chain, implying ϕ_t given ϕ_{t-1} is independent of all prior time points $t-1$*

$$p(\phi_t \mid \phi_{1:t-1}, y_{1:t-1}) = p(\phi_t \mid \phi_{t-1}).$$

A Markov modeling cliché is the past is independent of the future given the present, meaning $p(\phi_{t-1} \mid \phi_{t:T}, y_{t:T}) = p(\phi_{t-1} \mid \phi_t)$.

Property (Conditional independence of observations)**.** *The current measurement y_t given the current state ϕ_t is conditionally independent of the measurement and state histories, i.e.,*

$$p(y_t \mid \phi_{0:t}, y_{1:t-1}) = p(y_t \mid \phi_t).$$

In principle, we may use the above definitions and properties to compute the posterior distribution through the joint prior on the states and the likelihood of the measurements

$$\begin{aligned} p(\phi_{0:T}) &= p(\phi_0) \prod_{k=1}^{T} p(\phi_t \mid \phi_{t-1}), \\ p(y_{1:T} \mid \phi_{0:T}) &= \prod_{k=1}^{T} p(y_k \mid \phi_t). \end{aligned}$$

Through Bayes' rule,

$$\begin{aligned} p(\phi_{0:T} \mid y_{1:T}) &= \frac{p(y_{1:T} \mid \phi_{0:T})\, p(\phi_{0:T})}{p(y_{1:T})} \\ &\propto p(y_{1:T} \mid \phi_{0:T})\, p(\phi_{0:T}), \end{aligned}$$

but using this construction prohibits online forecasting and sampling can be difficult.

Instead, Bayesian filtering involves recursive computing that utilizes the Markovian conditional independence to design efficient algorithms for computation and sampling. Consider a state-space model defined through (2.1). The first step in the recursion is the formulation of the prior distribution or one-step-ahead predictive density for the latent state $p(\phi_{t-1} \mid y_{1:t-1})$ according to

$$p(\phi_t \mid \boldsymbol{y}_{1:t-1}) = \int p(\phi_t \mid \phi_{t-1})\, p(\phi_{t-1} \mid \boldsymbol{y}_{1:t-1})\, d\phi_{t-1}. \tag{2.2}$$

To show this, note that ϕ_t is conditionally independent of $\boldsymbol{y}_{1:t-1}$ given ϕ_{t-1}. Therefore,

$$\begin{aligned} p(\phi_t \mid \boldsymbol{y}_{1:t-1}) &= \int p(\phi_{t-1}, \phi_t \mid \boldsymbol{y}_{1:t-1})\, d\phi_{t-1} \\ &= \int p(\phi_t \mid \phi_{t-1}, \boldsymbol{y}_{1:t-1})\, p(\phi_{t-1} \mid \boldsymbol{y}_{1:t-1})\, d\phi_{t-1} \\ &= \int p(\phi_t \mid \phi_{t-1})\, p(\phi_{t-1} \mid \boldsymbol{y}_{1:t-1})\, d\phi_{t-1}. \end{aligned} \tag{2.3}$$

The filtering density then follows from Bayes' rule and the conditional independence of $\boldsymbol{y}_t$ and $\boldsymbol{y}_{1:t-1}$ given ϕ_t

$$p(\phi_t \mid \boldsymbol{y}_{1:t}) = \frac{p(\boldsymbol{y}_t \mid \phi_t, \boldsymbol{y}_{1:t-1})\, p(\phi_t \mid \boldsymbol{y}_{1:t-1})}{p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1})} = \frac{p(\phi_t \mid \boldsymbol{y}_{1:t-1})\, p(\boldsymbol{y}_t \mid \phi_t)}{p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1})}. \tag{2.4}$$

To normalization constant is the marginal likelihood or one-step-ahead predictive density. This is computed through

$$p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}) = \int p(\boldsymbol{y}_t \mid \phi_t)\, p(\phi_t \mid \boldsymbol{y}_{1:t-1})\, d\phi_t. \tag{2.5}$$

As $\boldsymbol{y}_t$ is conditionally independent of $\boldsymbol{y}_{1:t-1}$ given the state ϕ_t,

$$\begin{aligned} p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}) &= \int p(\boldsymbol{y}_t, \phi_t \mid \boldsymbol{y}_{1:t-1})\, d\phi_t \\ &= \int p(\boldsymbol{y}_t \mid \phi_t, \boldsymbol{y}_{1:t-1})\, p(\phi_t \mid \boldsymbol{y}_{1:t-1})\, d\phi_t \\ &= \int p(\boldsymbol{y}_t \mid \phi_t)\, p(\phi_t \mid \boldsymbol{y}_{1:t-1})\, d\phi_t. \end{aligned} \tag{2.6}$$

A central utility of state-space models is estimation and forecasting can be applied online as new data become available. However, in time series analysis there is often a fixed observation period for $t = 1, \ldots, T$. Inference may focus on retrodictive analysis of the behavior of the system to study the process and parameters. The backward recursive algorithm is used to compute the conditional distributions of ϕ_t given the full data $\boldsymbol{y}_{1:T}$. This recursion starts from the filtering distribution $p(\phi_T \mid \boldsymbol{y}_{1:T})$ and proceeds backwards. Conditional on $\boldsymbol{y}_{1:T}$, the state sequence $(\theta_0, \ldots, \phi_T)$ has backward transition distribution given through Bayes' rule as

$$p(\phi_t \mid \phi_{t+1}, \boldsymbol{y}_{1:T}) = \frac{p(\phi_{t+1} \mid \phi_t)\, p(\phi_t \mid \boldsymbol{y}_{1:t})}{p(\phi_{t+1} \mid \boldsymbol{y}_{1:t})}. \tag{2.7}$$

The smoothing distributions of ϕ_t given $\boldsymbol{y}_{1:T}$ can be computed according to the following backward recursion in t, starting from $p(\phi_T \mid \boldsymbol{y}_{1:T})$ since

$$p(\phi_t \mid \boldsymbol{y}_{1:T}) = p(\phi_t \mid \boldsymbol{y}_{1:t}) \int \frac{p(\phi_{t+1} \mid \phi_t)}{p(\phi_{t+1} \mid \boldsymbol{y}_{1:t})} p(\phi_{t+1} \mid \boldsymbol{y}_{1:T})\, d\phi_{t+1}. \tag{2.8}$$

There are two important conditional independence results necessary to prove the above. The first relies on the independence between ϕ_t and $Y_{t+1:T}$ given ϕ_{t+1}, while the second notes ϕ_{t+1} and $Y_{1:t}$ are conditionally independent given ϕ_t. This can be shown through DAG (2.2). Marginalizing $p(\phi_t, \phi_{t+1} \mid \boldsymbol{y}_{1:T})$ with respect to ϕ_{t+1} gives

$$\begin{aligned} p(\phi_t \mid \boldsymbol{y}_{1:T}) &= \int p(\phi_t, \phi_{t+1} \mid \boldsymbol{y}_{1:T})\, d\phi_{t+1} \\ &= \int p(\phi_t \mid \phi_{t+1}, \boldsymbol{y}_{1:T})\, p(\phi_{t+1} \mid \boldsymbol{y}_{1:T})\, d\phi_{t+1}. \end{aligned} \tag{2.9}$$

Bayes' rule and substitution then gives

$$\begin{aligned} p(\phi_t \mid \phi_{t+1}, \boldsymbol{y}_{1:T}) &= p(\phi_t \mid \phi_{t+1}, \boldsymbol{y}_{1:t}) \\ &= \frac{p(\phi_t \mid \boldsymbol{y}_{1:t})\, p(\phi_{t+1} \mid \phi_t, \boldsymbol{y}_{1:t})}{p(\phi_{t+1} \mid \boldsymbol{y}_{1:t})} \\ &= \frac{p(\phi_t \mid \boldsymbol{y}_{1:t})\, p(\phi_{t+1} \mid \phi_t)}{p(\phi_{t+1} \mid \boldsymbol{y}_{1:t})}. \end{aligned} \tag{2.10}$$

This results in the smoothing distribution of

$$\begin{aligned} p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:T}) &= \int p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{y}_{1:T}) \frac{p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{\phi}_t)\, p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t})}{p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{y}_{1:t})} d\boldsymbol{\phi}_{t+1} \\ &= p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t}) \int p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{\phi}_t) \frac{p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{y}_{1:T})}{p(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{y}_{1:t})} d\boldsymbol{\phi}_{t+1}. \end{aligned} \tag{2.11}$$

For a conjugate Gaussian state-space model, the smoothing recursion are available in closed form in the next section.

2.2.1 Conjugate Filtering and Smoothing

The prior and filtering expressions of (2.2) and (2.4) are available in closed form for certain classes of models. One such specification is a conjugate state-space model, utilized in Chapter 4. Let a time series of observations $\{\boldsymbol{y}_t, \ldots, \boldsymbol{y}_T\}$ be modeled with a Gaussian state-space form such that

$$\begin{aligned} \boldsymbol{y}_t \mid \boldsymbol{\phi}_t, v &\sim \mathcal{N}_n(\boldsymbol{F}_t \boldsymbol{\phi}_t, v\boldsymbol{V}_t) \\ \boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1}, v &\sim \mathcal{N}_p(\boldsymbol{G}_t \boldsymbol{\phi}_{t-1}, v\boldsymbol{W}_t). \end{aligned} \tag{2.12}$$

To start the computation, the correlation matrices $\boldsymbol{V}_t, \boldsymbol{W}_t$, as well as the transition matrices $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are assumed known or fixed. The scale parameter v is unknown and to be inferred along with the latent state $\boldsymbol{\phi}_{0:T}$. In Chapter 4, we relax the assumption of static v, along with $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$ known matrices. The parameter uncertainty in the model is then contained in the latent state vector and in the scaling parameter v. A convenient choice of prior for v^{-1} is conjugate normal-gamma family such that $v^{-1} \sim \mathcal{G}(n_0, d_0)$ and $\boldsymbol{\phi}_0 \mid v \sim \mathcal{N}(m_0, v\boldsymbol{M}_0)$, jointly denoted as $(\boldsymbol{\phi}_0, v) \sim \mathcal{NG}(m_0, \boldsymbol{M}_0, n_0, d_0)$. Since (2.2) holds for $t = 1$, assume it holds at time $t-1$ for the induction step so that

$$(\boldsymbol{\phi}_{t-1}, v) \mid \boldsymbol{y}_{1:t-1} \sim \mathcal{NG}(m_{t-1}, \boldsymbol{M}_{t-1}, n_{t-1}, d_{t-1}). \tag{2.13}$$

The prior for time t – also known as the one-step-ahead state predictive density – is then normal-gamma $\mathcal{NG}(\boldsymbol{a}_t, \boldsymbol{A}_t, n_{t-1}, d_{t-1})$, where

$$\boldsymbol{a}_t = \boldsymbol{G}_t \boldsymbol{m}_{t-1} \text{ and } \boldsymbol{A}_t = \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{W}_t. \tag{2.14}$$

As we supposed the prior $\boldsymbol{\phi}_{t-1}, \nu^{-1} \mid \boldsymbol{y}_{1:t-1} \sim \mathcal{NG}(m_{t-1}, \boldsymbol{M}_{t-1}, n_{t-1}, d_{t-1})$, by iterated conditional expectation, we have $\boldsymbol{\phi}_t \mid \nu^{-1}, \boldsymbol{y}_{1:t-1} \sim \mathcal{N}_p(\boldsymbol{a}_t, \nu \boldsymbol{A}_t)$, with $\boldsymbol{a}_t$ and $\boldsymbol{A}_t$ given by

$$\boldsymbol{a}_t = \mathbb{E}(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t-1}) = \mathbb{E}(\mathbb{E}(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1}, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) = \boldsymbol{G}_t \boldsymbol{m}_{t-1} \tag{2.15}$$

and

$$\begin{aligned} \nu \boldsymbol{A}_t &= \text{Var}(\boldsymbol{\phi}_t \mid \nu, \boldsymbol{y}_{1:t-1}) \\ &= \mathbb{E}(\text{Var}(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1} \nu, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) + \text{Var}(\mathbb{E}(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t-1}, \nu, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) \\ &= \nu[\boldsymbol{W}_t + \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top], \end{aligned} \tag{2.16}$$

from which it follows $\left(\boldsymbol{\phi}_t, \nu^{-1}\right) \mid \boldsymbol{y}_{1:t-1} \sim \mathcal{NG}(\boldsymbol{a}_t, \boldsymbol{A}_t, n_{t-1}, d_{t-1})$.

As shown in the previous section, the marginal likelihood or one-step-ahead predictive density $p(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1})$ is necessary to complete the filtering distribution. We first show $\boldsymbol{y}_t \mid \nu^{-1}, \boldsymbol{y}_{1:t-1} \sim \mathcal{N}_n(\boldsymbol{q}_t, \nu \boldsymbol{Q}_t)$. As above, we make use of iterated expectation through

$$\boldsymbol{q}_t = \mathbb{E}(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}) = \mathbb{E}(\mathbb{E}(\boldsymbol{y}_t \mid \boldsymbol{\phi}_t, \nu, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) = \mathbb{E}(\boldsymbol{F}_t \boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t-1}) = \boldsymbol{F}_t \boldsymbol{a}_t \tag{2.17}$$

and

$$\begin{aligned} \nu \boldsymbol{Q}_t &= \text{Var}(\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}) \\ &= \mathbb{E}(\text{Var}(\boldsymbol{y}_t \mid \boldsymbol{\phi}_t, \nu, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) + \text{Var}(\mathbb{E}(\boldsymbol{y}_t \mid \boldsymbol{\phi}_t, \nu, \boldsymbol{y}_{1:t-1}) \mid \boldsymbol{y}_{1:t-1}) \\ &= \nu[\boldsymbol{V}_t + \boldsymbol{F}_t \boldsymbol{A}_t \boldsymbol{F}_t^\top]. \end{aligned} \tag{2.18}$$

This implies $\boldsymbol{y}_t \mid \nu^{-1}, \boldsymbol{y}_{1:t-1} \sim \mathcal{N}_n(\boldsymbol{q}_t, \nu \boldsymbol{Q}_t)$, where $\boldsymbol{q}_t$ and $\boldsymbol{Q}_t$ are given above. Thus, we obtain $\left(\boldsymbol{y}_t, \nu^{-1}\right) \mid \boldsymbol{y}_{1:t-1} \sim \mathcal{NG}(\boldsymbol{q}_t, \boldsymbol{Q}_t, n_{t-1}, d_{t-1})$, so the corresponding marginal density of $\boldsymbol{y}_t \mid \boldsymbol{y}_{1:t-1}$ is Student-t with parameters $(\boldsymbol{q}_t, \boldsymbol{Q}_t d_{t-1}/n_{t-1}, 2n_{t-1})$.

We may now form the filtering distribution $p\left(\phi_t, \nu^{-1} \mid \boldsymbol{y}_{1:t}\right)$ through Bayes' rule of (2.4) to arrive at a posterior of form normal-gamma with parameters

$$\begin{aligned} m_t &= \boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{F}_t \boldsymbol{Q}^{-1} (\boldsymbol{y}_t - \boldsymbol{q}_t) \\ \boldsymbol{M}_t &= \boldsymbol{A}_t - \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{A}_t^\top \\ n_t &= n_{t-1} + \frac{n}{2} \\ d_t &= d_{t-1} + \frac{1}{2} (\boldsymbol{y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{y}_t - \boldsymbol{q}_t). \end{aligned} \tag{2.19}$$

The marginal filtering density of ν given $\boldsymbol{y}_{1:t}$ is inverse-gamma, with parameters (n_t, d_t), implying

$$\mathbb{E}(\nu \mid \boldsymbol{y}_{1:t}) = \frac{d_t}{n_t - 1}, \quad \operatorname{Var}(\nu \mid \boldsymbol{y}_{1:t}) = \frac{d_t^2}{(n_t - 1)^2 (n_t - 2)}. \tag{2.20}$$

Lastly, the marginal filtering density of the state is Student-t such that $\phi_t \mid \boldsymbol{y}_{1:t} \sim \mathcal{T}(m_t, \boldsymbol{M}_t d_t / n_t, 2n_t)$. To contrast the standard Kalman filter, were ν known, the filter would give the same point estimate of $\mathbb{E}(\phi_t \mid \boldsymbol{y}_{1:t}) = \boldsymbol{m}_t$, with covariance matrix $\operatorname{Var}(\phi_t \mid \boldsymbol{y}_{1:t}) = \nu \boldsymbol{M}_t$. Instead, the unknown value of ν is replaced by the conditional expectation $d_t/(n_t - 1)$. The larger uncertainty is reflected by a filtering density with thicker tails. To write the smoothing distribution of (2.11), note that $\left(\phi_T, \nu^{-1} \mid \boldsymbol{y}_{1:T}\right) \sim \mathcal{NG}(\boldsymbol{s}_T, \boldsymbol{S}_T, n_T, d_T)$ with $\boldsymbol{s}_T = m_T$ and $\boldsymbol{S}_T = \boldsymbol{M}_T$, and write $p\left(\phi_t, \nu^{-1} \mid \boldsymbol{y}_{1:T}\right) = p\left(\phi_t \mid \nu^{-1}, \boldsymbol{y}_{1:T}\right) p\left(\nu^{-1} \mid \boldsymbol{y}_{1:T}\right)$. The parameters can be computed using recursive formulas in (2.11) and are shown for $t = T-1, \ldots, 0$ to be

$$\begin{aligned} \boldsymbol{s}_t &= m_t + \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{s}_{t+1} - \boldsymbol{a}_{t+1}) \\ \boldsymbol{S}_t &= \boldsymbol{M}_t - \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{A}_{t+1} - \boldsymbol{S}_{t+1}) \boldsymbol{A}_{t+1}^{-1} \boldsymbol{G}_{t+1} \boldsymbol{M}_t. \end{aligned} \tag{2.21}$$

To prove, we again apply conditional expectation operations by way of

$$\begin{aligned} \boldsymbol{s}_t &= \mathbb{E}\left(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:T}\right) = \mathbb{E}\left(\mathbb{E}\left(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1}, \boldsymbol{y}_{1:T}\right) \mid \boldsymbol{y}_{1:T}\right) \\ \boldsymbol{S}_t &= \operatorname{Var}\left(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:T}\right) = \operatorname{Var}\left(\mathbb{E}\left(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1}, \boldsymbol{y}_{1:T}\right) \mid y_{1:T}\right) + \mathbb{E}\left(\operatorname{Var}\left(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1}, \boldsymbol{y}_{1:T}\right) \mid \boldsymbol{y}_{1:T}\right). \end{aligned} \tag{2.22}$$

This implies

$$\boldsymbol{\phi}_t, \boldsymbol{\nu}^{-1} \mid \boldsymbol{y}_{1:T} \sim \mathcal{NG}\left(\boldsymbol{s}_t, \boldsymbol{S}_t, n_T, d_T\right). \tag{2.23}$$

2.2.2 Hidden Markov Models

State-space models in which the state ϕ_t is discrete are termed hidden Markov models (HMMs). Chapter 5 concludes with a brief discussion on building an emulator based upon mixtures and HMMs to capture nonlinear behavior, so we review the details here. This framework is likely fruitful in developing more powerful surrogate models. In application, HMMs are used extensively across scientific modeling where outcomes might have distinct structural breaks. The dynamics of the series and the change points are thought determined by a latent Markov chain, meaning HMMs are a special class of the state-space framework where the latent process is discrete. As before, the inference objective is to learn about the hidden state and underlying variance components. For simplicity, assume the observed data $\boldsymbol{y}_{1:t}$ is generated by a mixture of two Gaussians such that

$$\boldsymbol{y}_t = \mu_{s_i} + \varepsilon_{s_i}, \varepsilon_{s_i} \sim \mathcal{N}(0, \Sigma_{s_i}), \tag{1}$$

where s_t is itself a random variable taking values in $\{1,2\}$. Assuming s_t is Markovian throughout time, the dynamics of s_t are governed through the Markov transition probabilities

$$\begin{aligned} P(s_t = 1 | s_{t-1} = 2) = p_{21},\ P(s_t = 2 | s_{t-1} = 1) = p_{12} \\ P(s_t = 1 | s_{t-1} = 1) = p_{11},\ P(s_t = 2 | s_{t-1} = 2) = p_{22}. \end{aligned} \tag{2.24}$$

The inferential goal is to learn these transition probabilities as well as recover the system parameters and latent states $s_{1:T}$. Denote the system parameters through

$$\boldsymbol{\theta} := \{\mu_1, \mu_2, \Sigma_1, \Sigma_2, p_{11}, p_{12}, p_{21}, p_{22}\}, \tag{2.25}$$

so the desired posterior distribution is $p(\boldsymbol{\theta}, s_{0:T}|\boldsymbol{y}_{1:t})$. Like in the previous derivations, the prior $p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})$, marginal likelihood $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1})$, and filtering distribution $p(\boldsymbol{s}_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta})$ are of central importance. The prediction equation first comes from integrating out all possible ways to arrive at s_t from the previous time step through the Chapman-Kolmogorov equation

$$\begin{aligned} p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) &= \int p(s_t, s_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_{t-1} \\ &= \int p(s_t|s_{t-1}, \boldsymbol{\theta}) p(s_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_{t-1}. \end{aligned} \tag{2.26}$$

Now the update (filter) step uses this prediction step through

$$\begin{aligned} p(s_t|\boldsymbol{y}_{1:t}, \boldsymbol{\theta}) &= \frac{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, s_t, \boldsymbol{\theta}) p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})}{\int p(y_t|s_t, \boldsymbol{\theta}) p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_t} \\ &= \frac{p(\boldsymbol{y}_t|s_t, \boldsymbol{\theta}) p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})}{p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta})}. \end{aligned} \tag{2.27}$$

The Bayesian filtering process then starts with an initial distribution for s_0 and iterates between the filter and update step for all time points. Finally, the posterior distribution of the model parameters is computed by integrating out the latent state at time t

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{y}_{1:t}) &\propto p(\boldsymbol{\theta}) \int p(\boldsymbol{y}_t, s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_t \\ &= p(\boldsymbol{\theta}) \int p(\boldsymbol{y}_t|s_t, \boldsymbol{\theta}) p(s_t|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_t \\ &= p(\boldsymbol{\theta}) \int p(\boldsymbol{y}_t|s_t, \boldsymbol{\theta}) \int p(s_t|s_{t-1}, \boldsymbol{\theta}) p(s_{t-1}|\boldsymbol{y}_{1:t-1}, \boldsymbol{\theta}) ds_{t-1} ds_t \end{aligned} \tag{2.28}$$

2.2.3 Exponential Family State-space Models

Gaussian state-space models can be generalized for modeling outcomes arising from the exponential family. For example, if $\boldsymbol{y}_t$ represents a binary characteristic over time, a Bernoulli distribution could be appropriate; if $\boldsymbol{y}_t$ are counts, the Poisson or negative binomial might be appropriate. In this way, generalized state-space models assume the conditional distribution $p(\boldsymbol{y}_t \mid \phi_t)$ of $\boldsymbol{y}_t$ given ϕ_t is a member of the exponential family, with natural parameter $\eta_t = \boldsymbol{F}_t\theta_t$. The state equation remains linear through $\theta_t = G_t\theta_{t-1} + w_t$. Inference for generalized state-space models presents computational difficulties solved by MCMC techniques (West et al., 1985). Further extensions to the methodology of Chapter 4 are possible for exponential family computer simulators.

2.2.4 Sampling and Inference

To build a sampling algorithm, we can write the joint distribution of $\phi_{0:T}$ given $\boldsymbol{y}_{1:T}$ as

$$p(\phi_{0:T} \mid \boldsymbol{y}_{1:T}) = \prod_{t=0}^{T} p(\phi_t \mid \phi_{t+1:T}, \boldsymbol{y}_{1:T}). \tag{2.29}$$

Sampling starts by drawing ϕ_T from $\mathcal{N}(m_T, C_T)$ and for $t = T-1, T-2, \ldots, 0$, recursively sampling ϕ_t from $p(\phi_t \mid \phi_{t+1:T}, \boldsymbol{y}_{1:T})$. Expression (2.11) showed $p(\phi_t \mid \phi_{t+1:T}, \boldsymbol{y}_{1:T}) = p(\phi_t \mid \phi_{t+1}, \boldsymbol{y}_{1:t})$, and (2.21 gave the distribution for the Gaussian state-space model of $\mathcal{N}(\boldsymbol{s}_t, \boldsymbol{S}_t)$, with

$$\begin{aligned} \boldsymbol{s}_t &= \boldsymbol{m}_t + \boldsymbol{M}_t\boldsymbol{G}_{t+1}^{\top}\boldsymbol{A}_{t+1}^{-1}(\phi_{t+1} - \boldsymbol{a}_{t+1}) \\ \boldsymbol{S}_t &= \boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^{\top}\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t \end{aligned} \tag{2.30}$$

Therefore, having obtained $(\phi_{t+1}, \ldots, \phi_T)$, the next iteration draws ϕ_t from a $\mathcal{N}(\boldsymbol{s}_t, \boldsymbol{S}_t)$. The FFBS algorithm is summarized below. Chapter 4 relies on this sampling method.

1. Run Kalman filter.
2. Draw $\phi_T \sim \mathcal{N}(m_T, C_T)$.
3. For $t = T-1, \ldots, 0$, draw $\phi_t \sim \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{S}_t)$.

2.3 Mechanistic State-space Parameterizations

In the previous section, we described state-space methodology, computation, and Bayesian inference. This section discusses mechanistic parameterizations of such models. These are specifically inspired through modeling with algebraic equations that can be discretized, such as simple diffusion processes, but the techniques are generally applicable. Diffusion models are well-studied in the mathematical biology and physics literature (Keeling and Rohani, 2008) but are less utilized in statistical practice and application. Therefore, there exist opportunities to connect these models to data through the use of Bayesian state-space methods and ultimately Gaussian processes through the work of Chapter 4. An application of methodology proposed within this dissertation concerns space-time dynamics of the Laplacian in 2-dimensional space, so we focus on a simple discretization to motivate the dynamics. Let f be a function $f : \mathbb{R}^3 \to \mathbb{R}$ such that

$$\begin{aligned} \frac{\partial f_t(x,y)}{\partial t} &= \alpha \frac{\partial^2}{\partial x^2} f_t(x,y) + \alpha \frac{\partial^2}{\partial y^2} f_t(x,y) \\ &= \alpha \nabla^2 f_t(x,y). \end{aligned} \tag{2.31}$$

The Laplacian term ∇^2 is introduced to model the local diffusion throughout space, where ∇ is shorthand for the rate of change across space. This implies ∇^2 represents the change in the rate of change. The inclusion of these spatial derivatives mimics the diffusion of a substance across the spatial domain. In our setting, these will motivate spatially-varying coefficient model parameterizations.

2.3.1 Finite Difference Methods

As the spatial dynamics are induced through the Laplacian ∇^2, this subsection will establish a finite-difference approximation over a discretization of space to motivate the parameterization. To gain intuition, a function $p_t : \mathbb{R}^2 \to \mathbb{R}$ is the solution to the heat equation over 1-dimensional space and time if

$$\frac{\partial p_t(x)}{\partial t} = \alpha \frac{\partial^2 p_t(x)}{\partial x^2} = \alpha \nabla^2 p_t(x). \tag{2.32}$$

Approximating the solution to (2.32) starts with the forward and backward difference approximation of the derivative

$$\begin{aligned}\frac{\partial p_t(x)}{\partial x} &= \lim_{\delta x \to 0} \frac{p_t(x+\delta x) - p_t(x)}{\delta x} \\ \frac{\partial p_t(x)}{\partial x} &= \lim_{\delta x \to 0} \frac{p_t(x) - p_t(x-\delta x)}{\delta x}.\end{aligned} \tag{2.33}$$

Adding these together gives the central difference approximation of

$$\frac{\partial p_t(x)}{\partial x} = \lim_{\delta x \to 0} \frac{p_t(x+\delta x) - p_t(x-\delta x)}{2\delta x}. \tag{2.34}$$

For notational simplicity, define $\Delta x := 2\delta x$ so that (2.34) becomes

$$\frac{\partial p_t(x)}{\partial x} = \lim_{\Delta x \to 0} \frac{p_t(x+\frac{1}{2}\Delta x) - p_t(x-\frac{1}{2}\Delta x)}{\Delta x}. \tag{2.35}$$

Applying the central difference approximation twice to approximate the Laplacian $\frac{\partial^2 p_t(x)}{\partial x^2}$ gives

$$\begin{aligned}\frac{\partial^2 p_t(x)}{\partial x^2} &= \partial_x \lim_{\Delta x \to 0} \frac{p_t(x+\frac{1}{2}\Delta x) - p_t(x-\frac{1}{2}\Delta x)}{\Delta x} \\ &= \lim_{\Delta x \to 0} \frac{\frac{p_t(x+\Delta x) - p_t(x)}{\Delta x} - \frac{p_t(x) - p_t(x-\Delta x)}{\Delta x}}{\Delta x} \\ &\approx \frac{p_t(x+\Delta x) - 2p_t(x) + p_t(x-\Delta x)}{(\Delta x)^2} \text{ for small } \Delta x.\end{aligned} \tag{2.36}$$

For simplicity, let $\Delta x = \Delta y$ and apply the approximation in (2.36) to (2.31) to get

$$\begin{aligned}\nabla^2 f_t(x,y) \approx \frac{1}{(\Delta x)^2}\Big[&f_t(x+\Delta x,y) - 2f_t(x,y) + f_t(x-\Delta x,y) \\ &+ f_t(x,y+\Delta y) - 2f_t(x,y) + f_t(x,y-\Delta y)\Big].\end{aligned} \tag{2.37}$$

In this way, on a 2-dimensional grid with spacing of $\Delta x = \Delta y$, the PDEs are reduced to a coupled set of ODEs on the lattice. This discretization is visualized below in Figure 2.1.

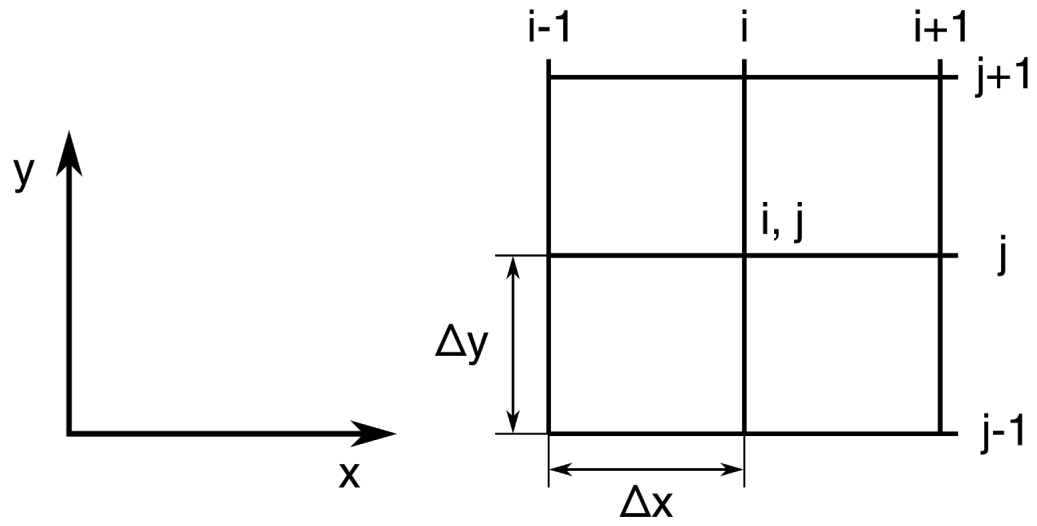


Figure 2.1: Discretization of partial differential equation in 2-dimensional space

As a concrete example to connect to Chapter 4, this discretization can be substituted into expression (4.23) to arrive at the approximation

$$\begin{aligned}
\frac{\partial S_t(x,y)}{\partial t} &\approx -\beta \frac{S_t(x,y)I_t(x,y)}{N} + \frac{1}{(\Delta x)^2}\left[S_t(x \pm \Delta x, y) + S_t(x, y \pm \Delta y) - 4S_t(x,y)\right] \\
\frac{\partial I_t(x,y)}{\partial t} &\approx \beta \frac{S_t(x,y)I_t(x,y)}{N} - \gamma I_t(x,y) + \frac{1}{(\Delta x)^2}\left[I_t(x \pm \Delta x, y) + I_t(x, y \pm \Delta y) - 4I_t(x,y)\right] \qquad (2.38) \\
\frac{\partial R_t(x,y)}{\partial t} &\approx \gamma I_t(x,y) + \frac{1}{(\Delta x)^2}\left[R(x \pm \Delta x, y) + R_t(x, y \pm \Delta y) - 4R_t(x,y)\right].
\end{aligned}$$

The numerical solution is then obtained by looping over the grid coordinates and solving the coupled ODEs numerically. In `R`, such a loop might take the form

```
diffusion.sir <- function(S,I,R) {
  for(i in 2:(gridsize-1)){
    for(j in 2:(gridsize-1)){
      dS[i,j] = -beta*S[i,j]*I[i,j]/N
        +alpha1/d^2*(S[i+1,j]+S[i-1,j]+S[i,j+1]+S[i,j-1]-4*S[i,j])
      dI[i,j] = beta*S[i,j]*I[i,j]/N-gamma*I[i,j]
```

```
        +alpha2/d^2*(I[i+1,j]+I[i-1,j]+I[i,j+1]+I[i,j-1]-4*I[i,j])
      dR[i,j] = gamma*I[i,j]
        +alpha3/d^2*(I[i+1,j]+I[i-1,j]+I[i,j+1]+I[i,j-1]-4*R[i,j])
    }
  }
}
```

Dirichlet boundary conditions are used to set the boundary to zero, but in general boundary effects may be modeled as well (Wikle et al., 2019). In conjunction with the `R` package `deSolve`, the above function can be used to numerically solve the PDEs. This hints at the computational complexities involved for large spatial domains and complicated systems of PDEs. In the next section, we discuss how this idea of discretization is central to parameterizing informative, mechanistic state-space models.

2.3.2 Connection to State-space Models

The previous finite difference approximations show how the lattice induces neighbor effects. This motivates a lagged-nearest-neighbor (LNN) model for the latent process (Wikle and Hooten, 2010). The LNN parameterization can be motivated by many mechanistic models, such as those suggested by discretization of integro-differential or PDEs, but we consider functions of the form (2.31). First consider a spatiotemporal state-space model on a 2-dimensional spatial domain of the form

$$\begin{aligned} \boldsymbol{y}_t(x,y) &= \boldsymbol{F}_t(x,y)\boldsymbol{\phi}(x,y) + \boldsymbol{\varepsilon}_t(x,y),\ \boldsymbol{\varepsilon}_t(x,y) \sim \mathcal{N}(0, \boldsymbol{V}_t) \\ \boldsymbol{\phi}_t(x,y) &= \boldsymbol{G}(x,y,\boldsymbol{\theta})\boldsymbol{\phi}(x,y) + \boldsymbol{\tau}_t(x,y),\ \boldsymbol{\tau}_t(x,y) \sim \mathcal{N}(0, \boldsymbol{W}_t) \end{aligned} \tag{2.39}$$

We now discuss how $\boldsymbol{G}_t(\boldsymbol{\theta})$ can induce mechanistically-inspired process evolution. Consider the basic 2-dimensional heat equation (2.31) of the previous subsection but generalize to allow for

spatially-varying diffusion coefficients through

$$\frac{\partial z(x,y)}{\partial t} = \alpha_1(x,y)\frac{\partial^2 z(x,y)}{\partial x^2} + \alpha_2(x,y)\frac{\partial^2 z(x,y)}{\partial y^2}. \tag{2.40}$$

The coefficients α_1, α_2 are diffusion coefficients that control the rate of spread across space. If we consider this process on a regular 2-dimensional grid and employ a standard centered finite difference like in (2.37) and a forward difference in time, we obtain a LNN model of the form

$$\begin{aligned}\phi_t(x,y) = {} & \theta_1(x,y)\phi_{t-\Delta_t}(x,y) + \theta_2(x,y)\phi_{t-\Delta_t}(x+\Delta_x,y) \\ & + \theta_3(x,y)\phi_{t-\Delta_t}(x-\Delta_x,y) + \theta_4(x,y)\phi_{t-\Delta_t}(x,y+\Delta_y) \\ & + \theta_5(x,y)\phi_{t-\Delta_t}(x,y-\Delta_y)\end{aligned} \tag{2.41}$$

where Δ_t is a time-discretization constant, Δ_x and Δ_y are spatial-discretization constants, and the θ's are defined by using (2.37)

$$\begin{aligned}\theta_1(x,y) &= \left[\frac{-2\alpha_1(x,y)\Delta_t}{\Delta_x^2} + \frac{-2\alpha_2(x,y)\Delta_t}{\Delta_y^2}\right] + 1, \\ \theta_2(x,y) &= \frac{\alpha_1(x+\Delta_x,y)\Delta_t}{4\Delta_x^2} - \frac{\alpha_1(x-\Delta_x,y)\Delta_t}{4\Delta_x^2} + \frac{\alpha_1(x,y)\Delta_t}{\Delta_x^2} \\ \theta_3(x,y) &= \frac{-\alpha_1(x+\Delta_x,y)\Delta_t}{4\Delta_x^2} + \frac{\alpha_1(x-\Delta_x,y)\Delta_t}{4\Delta_x^2} + \frac{\alpha_1(x,y)\Delta_t}{\Delta_x^2} \\ \theta_4(x,y) &= \frac{\alpha_2(x,y+\Delta_y)\Delta_t}{4\Delta_y^2} - \frac{\alpha_2(x,y-\Delta_y)\Delta_t}{4\Delta_y^2} + \frac{\alpha_2(x,y)\Delta_t}{\Delta_y^2} \\ \theta_5(x,y) &= \frac{-\alpha_2(x,y+\Delta_y)\Delta_t}{4\Delta_y^2} + \frac{\alpha_2(x,y-\Delta_y)\Delta_t}{4\Delta_y^2} + \frac{\alpha_2(x,y)\Delta_t}{\Delta_y^2}.\end{aligned} \tag{2.42}$$

The finite-difference discretization on a 2-dimensional equally-spaced lattice then leads to an LNN autoregressive specification for the latent process in (2.39) through

$$\phi_t(x,y) = \boldsymbol{G}(x,y,\boldsymbol{\theta})\phi_{t-1}(x,y) + \tau_t(x,y),\ \tau_t(x,y) \sim \mathcal{N}(0,\boldsymbol{W}_t), \tag{2.43}$$

where $\boldsymbol{G}(x, y, \boldsymbol{\theta})$ is a block diagonal matrix containing the elements of $\boldsymbol{\theta}$. If the diffusion coefficients are assumed spatially homogeneous, these expressions in (2.42) simplify to

$$\theta_1 = 1,\ \theta_2 = \frac{\alpha_1 \Delta_t}{\Delta_x^2},\ \theta_3 = \frac{\alpha_1 \Delta_t}{\Delta_x^2},\ \theta_4 = \frac{\alpha_2 \Delta_t}{\Delta_y^2},\ \theta_5 = \frac{\alpha_2 \Delta_t}{\Delta_y^2}. \tag{2.44}$$

In this way, we lose a spatiotemporally-varying state transition matrix in favor of a more simplistic latent evolution process.

This section showed how the finite differences suggest that neighbors of spatial location (x, y) at the previous time of play a role in the time dynamics and spatially-varying process parameters. If we stack the observed data over space, it can be shown the ensuing transition operator is a large sparse matrix with symmetry determined if diffusion coefficients vary with space. This suggests future opportunities for dimensionality reduction (Wikle et al., 2019).

2.4 Emulation and Calibration with Gaussian Processes

In the previous section, we described how algebraic mechanisms can inspire parameterizations of a state-space model. However, this is not always possible for computer models or simulations, since there may be no explicit algebraic form motivating the dynamics. Even if an algebraic representation exists, it might be too difficult to inspire a state-space parameterization. In such a setting, we define a computer model to be a general one-to-one functional relationship with some mechanistic and typically low dimensional parameter space. Various potential constraints on the computer model can be summarized below.

- the computer is costly to evaluate, precluding extensive function calls to understand output dynamics
- lack of analytic form, rendering the computer model an expensive "black box" with no gradient information or convexity assumptions
- limited function calls prohibit Monte Carlo analysis for calibration

Bayesian approaches to this problem rely on modeling the computer/functional relationship probabilistically based on limited data. This circumvents evaluating the expensive underlying computer model many times. As a motivating example within the biological community, Farah et al. (2014) successfully trained a Gaussian process emulator to capture behavior of an underlying expensive epidemiological SEIR model to enable efficient parameter calibration. In the machine learning community, modeling with surrogate functions to solve difficult optimization problems is an active and fruitful field of research (Turner et al., 2021; Garnett, 2022). The purpose of this section is to further describe the convenient functional form by which it is common to probabilistically model the expensive system.

2.4.1 Statistical Emulation

The Bayesian analysis of computer experiments through use of Gaussian processes is mature, and early seminal works of Kennedy and O'Hagan (2001) and Oakley et al. (2004) provide readable introductions. We summarize the main results and modeling ideas in this section, with extensions to high-dimensional spatiotemporal output developed in Chapter 4. To probabilistically model the expensive process, the computer is treated as an unknown function. Thus modeling with Gaussian processes is natural. Let the computer be represented by $f : \mathbb{R}^d \to \mathbb{R}$, which maps an input vector $\boldsymbol{x} = (x_1, ..., x_d)^\top$ to a real-valued output $y(\boldsymbol{x})$. Modeling f with a Gaussian process involves specifying a mean function $\mu(\cdot)$ and kernel function $C(\cdot, \cdot; \boldsymbol{\theta})$, as any finite realization $\boldsymbol{y}(\mathcal{X}) = \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_n)\}$ is a n-dimensional multivariate Gaussian. This often offers improved computational tractability over the expensive simulator, as prediction amounts to standard Gaussian conditioning identities and linear algebra. It is often convenient to assume a squared exponential correlation function, but a general and widely used class is provided by the Matérn covariance function which, for two points at a distance d is defined by

$$C(d; \nu, \rho, \sigma^2) = \sigma^2 \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(2\sqrt{\nu} \frac{d}{\rho} \right)^{\nu} K_\nu \left(2\sqrt{\nu} \frac{d}{\rho} \right), \tag{2.45}$$

K_ν is the modified Bessel function of the second kind and $\rho, \nu, \sigma^2 > 0$. The squared exponential covariance function is then a special case as $\nu \to \infty$ in (2.45). We use the squared exponential throughout Chapter 4. A Gaussian process model given a mean and covariance functions is written

$$y(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\cdot), C(\cdot;\boldsymbol{\theta})). \tag{2.46}$$

In later chapters, we generalize to functions with output in $\mathbb{R}^p$. Once the Gaussian process hyperparameters are fixed, parameter calibration makes use of the cheap statistical model to fit to observed data. From standard identities, we can form the conditional distribution for a new input $\boldsymbol{x}^*$, in effect "learning" how the computer model behaves for unknown input values. These conditional identities result in a Gaussian predictive distribution $p\big(y(\boldsymbol{x}^*)|\boldsymbol{y}(\mathcal{X})\big) \sim \mathcal{N}(\bar{\mu}(\boldsymbol{x}^*), \bar{\sigma}^2(\boldsymbol{x}^*))$ with mean $\bar{\mu}(\boldsymbol{x}^*) = \mu(\cdot) + \boldsymbol{\gamma}^\top \boldsymbol{C}(\mathcal{X};\boldsymbol{\beta})^{-1}(\boldsymbol{y}(\mathcal{X}) - \mu(\cdot)\mathbf{1}_n)$ and variance $\bar{\sigma}^2(\boldsymbol{x}^*) = C(\boldsymbol{x}^*, \boldsymbol{x}^*;\boldsymbol{\beta}) - \boldsymbol{\gamma}(\boldsymbol{x}^*)^\top \boldsymbol{C}(\mathcal{X};\boldsymbol{\beta})^{-1}\boldsymbol{\gamma}(\boldsymbol{x}^*)$. We define $\boldsymbol{\gamma}(\boldsymbol{x}^*) = \big(C(\boldsymbol{x}_i, \boldsymbol{x}^*;\boldsymbol{\beta})\big)^\top$ for $i = 1, \ldots, n$ and $\boldsymbol{C}(\mathcal{X};\boldsymbol{\beta})$ to indicate $C(\cdot;\boldsymbol{\beta})$ applied to $\mathcal{X}$. These predictive equations imply the Gaussian process interpolates at the input points as the variance shrinks to zero.

2.4.2 Surrogate Model Calibration

In the emulation/calibration problem, if f is treated as an unknown function, the goal becomes to estimate f based upon limited design runs as well as infer the parameter setting that best connects to observed data. Once the hyperparameter of f are estimated, we may predict computer model output at $\boldsymbol{x}^*$ through the predictive distribution f^*. When observing field data, common in practice is to add a discrepancy/model bias term δ to account for model misspecification, either through emulator or computer model inadequacy. We discuss this model extension in Chapter 4. The Bayesian hierarchical calibration model is

$$\begin{aligned} z &= f^*(\boldsymbol{x}^*;\boldsymbol{\theta}) + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2) \\ f(\boldsymbol{x};\boldsymbol{\theta}) &\sim \mathcal{GP}(\mu, C(\cdot;\boldsymbol{\theta})), \end{aligned} \tag{2.47}$$

where f^* is the posterior predictive distribution of the process f. If this emulator model includes time and computer model input in $\boldsymbol{x}$, as the series length grows, this becomes computationally infeasible. This problem is exacerbated in the spatiotemporal setting, where space is also included in the input. This is the focus of Chapter 4. The two necessary data objects for the calibration step are the observed data and the designed simulator runs. For any fixed realization of design points, say $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we have the outcome vector $\boldsymbol{y} = \{y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n)\}$. Then the realized version of (2.47) is Gaussian such that

$$\begin{aligned} z &= f^*(\boldsymbol{x}^*; \boldsymbol{\theta}) + \varepsilon,\ \varepsilon \sim \mathcal{N}(0, \sigma^2) \\ f^*(\boldsymbol{x}^*; \boldsymbol{\theta}) &\sim \mathcal{N}(\widetilde{\mu}(\boldsymbol{x}^*), \widetilde{\sigma}^2(\boldsymbol{x}^*)), \end{aligned} \tag{2.48}$$

with

$$\begin{aligned} \bar{\mu}(\boldsymbol{x}^*) &= \mu + \boldsymbol{\gamma}(\boldsymbol{x}^*)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{y}(\mathcal{X}) - \mu \mathbf{1}_n) \\ \bar{\sigma}^2(\boldsymbol{x}^*) &= C(\boldsymbol{x}^*, \boldsymbol{x}^*; \boldsymbol{\beta}) - \boldsymbol{\gamma}(\boldsymbol{x}^*)^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\gamma}(\boldsymbol{x}^*). \end{aligned} \tag{2.49}$$

$\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix generated through $C(\cdot; \boldsymbol{\theta})$ applied to $\mathcal{X}$ and $\boldsymbol{\gamma}(\boldsymbol{x}^*) = \big(C(\boldsymbol{x}_i, \boldsymbol{x}^*; \boldsymbol{\beta})\big)^\top$ for $i = 1, \ldots, n$. We may collect the observed field data z and simulation runs $\boldsymbol{y}$ to model the emulation/calibration process jointly. Assuming the observed data arises from f at some unknown "optimal" parameter setting $\boldsymbol{x}^*$, of which we wish to infer, the mean vector is $\boldsymbol{\omega} = (\tilde{\mu}(\boldsymbol{x}^*), \mu \mathbf{1}_n)^\top$. The covariance structure $\boldsymbol{\Omega}$ is formed through an additive combination of $\boldsymbol{\Sigma}$ and the observational variance σ^2. The ij-th element of Ω is given by the $(n+1) \times (n+1)$,

$$\boldsymbol{\Omega} = \begin{pmatrix} C(\boldsymbol{x}^*, \boldsymbol{x}^*; \boldsymbol{\beta}) + \sigma^2 & \boldsymbol{\gamma}^\top(\boldsymbol{x}^*) \\ \boldsymbol{\gamma}(\boldsymbol{x}^*) & \boldsymbol{\Sigma} \end{pmatrix}, \tag{2.50}$$

where $\boldsymbol{\gamma}(\boldsymbol{x}^*) = \big(C(\boldsymbol{x}_1, \boldsymbol{x}^*; \boldsymbol{\beta}), \ldots, C(\boldsymbol{x}_n, \boldsymbol{x}^*; \boldsymbol{\beta})\big)^\top$. To build the full model, let $\boldsymbol{y}^* = (z, \boldsymbol{y})^\top$ so the likelihood is

$$p(\boldsymbol{y}^* | \mathcal{X}, \boldsymbol{x}^*, \boldsymbol{\theta}, \mu, \sigma^2) = |\Omega|^{-1/2} \exp\left(-\frac{1}{2} (\boldsymbol{y}^* - \boldsymbol{\omega})^\top \boldsymbol{\Omega}^{-1} (\boldsymbol{y}^* - \boldsymbol{\omega}) \right). \tag{2.51}$$

The resulting posterior is proportional to

$$p(\boldsymbol{x}^*, \mu, \boldsymbol{\theta}, \sigma^2 | \boldsymbol{y}^*) \propto p(\boldsymbol{y}^* | \mathcal{X}, \boldsymbol{x}^*, \mu, \boldsymbol{\theta}, \sigma^2) p(\boldsymbol{x}^*, \mu, \boldsymbol{\theta}, \sigma^2). \tag{2.52}$$

Once priors are specified, sampling may proceed. However, this formulation becomes computational intractable for large spatiotemporal problems. Additionally, it is often more desirable to first build the emulation module and then calibrate to observed field data, rather than estimating the full probability model jointly. We expand on this in Chapter 4.

CHAPTER 3

A Mechanistic Model of Early Covid-19 Infection Dynamics in NYC

3.1 Introduction

The global Covid-19 pandemic has underscored the importance of mathematical and statistical models in understanding disease dynamics, assessing policy efficacy, and examining counterfactual scenarios to formulate thorough cost-benefit analyses. Lockdown measures can have drastic impact on individual well-being as well as society and the economy at large Bonaccorsi et al. (2020), Cutler and Summers (2020). Therefore, a retrospective study of Covid-19 lockdown and mitigation measures can help policymakers and public health officials understand to what end such efforts were effective. The formulation of a mechanistic compartmental model is a pathway towards such goals. In this article, we review compartmental model methodology, construct our new Bayesian hierarchical model, and discuss numerical methods relevant for implementation and fitting to real-world data. The new compartmental model is a simple modification of the classical susceptible-infectious-removed (SIR) model and enables a mechanistic correspondence between smartphone mobility data and infection dynamics. This can provide evidence of how reduced mobility due to early lockdowns or mitigation measures within New York City influenced Covid-19 spread dynamics. Case count data is obtained from the official website of the City of New York, available at `https://www1.nyc.gov/site/doh/covid/covid-19-data.page`. Population transit mobility data is obtained from `https://covid19.apple.com/mobility` and consists of anonymized Apple iPhone transit usage reported as a percent relative to baseline. Starting from the day after Governor

Andrew Cuomo declared a state of emergency in New York State on March 7th, 2020, both the raw case count and transit mobility time series are presented below. Our end goal is then to establish a relationship between the two series shown in Figure 3.1.

3.2 Background

Mathematical modeling in epidemiology has a long history, famously dating back to the eighteenth century with the work of Bernoulli Dietz and Heesterbeek (2002) or the mid-nineteenth century through John Snow's modeling of the cholera outbreak in London Shiode et al. (2015). However, at the turn of the early twentieth century, mathematical epidemiology turned to the modern theory of dynamical systems analysis to understand outbreak evolution. In this section, we review the popular SIR model and subsequent mathematical analysis used to glean both qualitative and quantitative understanding of the dynamical system. This simple framework provides the necessary foundation for more complicated compartmental models with more population states. For examples of other compartment models designed to study early Covid-19 outbreak dynamics, see Hao et al. (2020) or Wang et al. (2020a).

3.2.1 SIR Compartmental Model

In modeling population-level data with a compartmental system, the population is typically subdivided into separate homogeneous groups. Here we focus on reviewing the simple SIR model developed in 1927 by A. G. McKendrick and W. O. Kermack to model a plague outbreak in Bombay Kermack and McKendrick (1927). In such a model, the population is divided into susceptible (S), infectious (I), and removed (R) groups. Individuals then progress through the various states at certain rates over time. The mathematical description of this changing system is the coupled set of ordinary differential

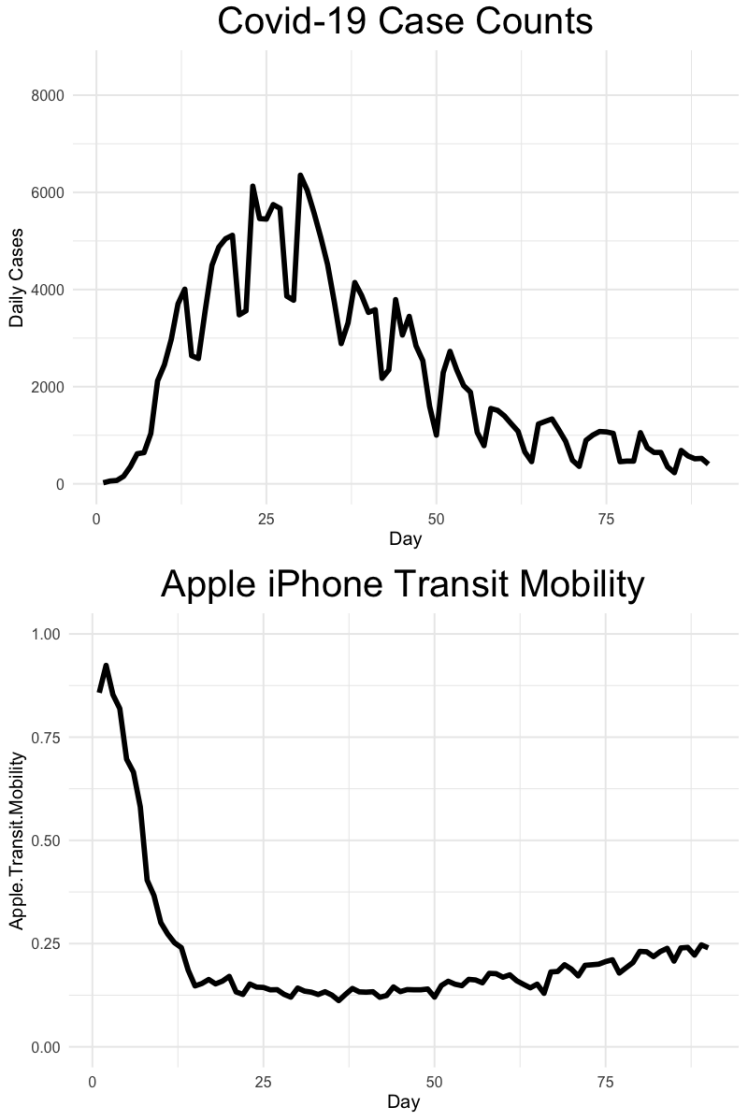


Figure 3.1: Raw daily Covid-19 case counts and cell phone Mobility as a percentage of nominal movement within New York City

equations

$$\begin{cases} \dfrac{dS(t)}{dt} &= -\beta S(t)I(t)/N \\ \dfrac{dI(t)}{dt} &= \beta S(t)I(t)/N - \gamma I(t) \\ \dfrac{dR(t)}{dt} &= \gamma I(t). \end{cases} \tag{3.1}$$

The progression of the disease throughout the population depends upon the contact rate between susceptible and infectious individuals, the probability of transmission upon contact, and the prevalence of disease. To mathematically capture these factors and express the rate of new infections, let λ be defined as a per capita contact rate among individuals. In this way, $\lambda S(t)$ will give the average number of susceptible contacts over time. Now let p be the probability that a contact results in a new infection. Finally, the prevalence of the disease at time t is by definition $I(t)/N$. Combining these terms gives $p\lambda S(t)I(t)/N$ as the incidence rate. In defining the *effective contact rate* β as the product of the per capita contact rate λ and transmission probability p, the necessary form in equation (3.1) is recovered.

The remaining parameter γ is interpreted by considering that $1/\gamma$ is the average sojourn time of an individual within compartment I. It should also be noted the population is fixed throughout time since $N = S + I + R$. This can alternatively be seen, since adding the terms in (3.1) gives zero. With meaning associated to the compartmental parameters, we next turn to an overview of the mathematical analysis involved in analyzing the basic SIR dynamical system.

3.2.2 Linear Stability Analysis and the Basic Reproductive Number

The system of differential equations in (3.1) are nonlinear, arising from the term $S(t)I(t)$. Linear stability analysis is the workhorse to understand the behavior of nonlinear dynamical systems and has a fundamental connection to the *basic reproductive number* $\mathcal{R}_0$, popularized recently through media coverage of the Covid-19 pandemic. $\mathcal{R}_0$ is defined roughly to be the expected number of subsequent infections resulting from a single infected individual. In this section, we briefly review linear stability

analysis and make the connection to $\mathcal{R}_0$. In the next section, we highlight the computation of $\mathcal{R}_0$ for a general class of compartmental models.

A steady state or equilibrium of a dynamical system is a point $\boldsymbol{x}^*$ where the system of differential equations evaluated at $\boldsymbol{x}^*$ is zero for all t. In this way, compartmental contents within the system are not changing over time. In the SIR model of (3.1), an important steady state is the so-called disease-free equilibrium of $\{(S^*,0,0) : S^* \geq 0\}$. A natural subsequent question is the behavior of the system around small perturbations of the equilibrium. In computing the Jacobian about such a point, we linearize and are afforded tractable analysis. A heuristic justification arises from considering a Taylor expansion of the system about the disease-free equilibrium and ignoring high-order terms, since the perturbation is assumed small. After dropping the explicit dependence on time t from equation (3.1) to avoid clutter, the Jacobian of the system is computed as

$$\boldsymbol{J} = \begin{pmatrix} \frac{\partial \dot{S}}{\partial S} & \frac{\partial \dot{S}}{\partial I} & \frac{\partial \dot{S}}{\partial R} \\ \frac{\partial \dot{I}}{\partial S} & \frac{\partial \dot{I}}{\partial I} & \frac{\partial \dot{I}}{\partial R} \\ \frac{\partial \dot{R}}{\partial S} & \frac{\partial \dot{R}}{\partial I} & \frac{\partial \dot{R}}{\partial R} \end{pmatrix} = \begin{pmatrix} -\beta I/N & -\beta S/N & 0 \\ \beta I/N & \beta S/N - \gamma & 0 \\ 0 & \gamma & 0 \end{pmatrix}. \tag{3.2}$$

Through elementary matrix operations, $\boldsymbol{J}$ can be transformed to a triangular matrix. The eigenvalues are then found from inspection. Evaluating the Jacobian $\boldsymbol{J}$ at $(S^*,0,0)$ to linearize about the steady state results in eigenvalues of $\lambda_1 = 0$ and $\lambda_2 = \beta S^*/N - \gamma$. The sign of these eigenvalues then determine the stability of the equilibrium point. The eigenvalue of λ_1 is ignored, as it corresponds to a line of equilibrium values S^*. In this way, the second eigenvalue of λ_2 is of main interest. If $N/S^* < \beta/\gamma$, then $\lambda_2 > 0$ and the steady state is unstable; otherwise it is stable. In epidemic terms, an outbreak occurs if the disease-free equilibrium is unstable. This ratio β/γ acts as a bifurcation parameter in determining if an outbreak will occur and is thus afforded the fancy title of basic reproductive number. Letting the equilibrium point S^* be the population size so that $S^* = N$, a simple relation emerges: if $\mathcal{R}_0 > 1$ the disease continues to spread but dies out otherwise. The effective reproductive number $\mathcal{R}_t$ then extends $\mathcal{R}_0$ by accounting for a changing susceptible population over time and is

defined as $\mathcal{R}_t := \mathcal{R}_0 S(t)/N$. A general method to compute $\mathcal{R}_0$ for more elaborate compartmental models will be discussed in the next subsection.

3.2.3 Spectral Radius or Next Generation Matrix Method

For general compartment models that extend the simplistic SIR framework, computing the basic reproductive number can be difficult. Diekmann et al. (2009) and Heffernan et al. (2005) describe a general method to compute $\mathcal{R}_0$ called the *Next Generation Matrix* or *Spectral Radius Method*, which we briefly review and compute for the SIR model.

Let $d\boldsymbol{X}(t)/dt$ represent a general coupled system of differential equations describing a compartmental model with n components and m infectious states. Define a vector-valued $\boldsymbol{F}(\boldsymbol{X}(t))$ to be a function where each component specifies flow rate into one of the respective m infected compartments. Similarly, define a function $\boldsymbol{V}(\boldsymbol{X}(t))$ where each component specifies the flow rate out of a respective infectious compartment.

Next, $\boldsymbol{F}$ and $\boldsymbol{V}$ are linearized about the disease-free equilibrium point by computing the Jacobian. Diekmann et al. (2009) prove the Jacobian with respect to each infectious state will take the form

$$\boldsymbol{J}(\boldsymbol{F}) = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \text{ and } \boldsymbol{J}(\boldsymbol{V}) = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{0} \\ * & * \end{pmatrix}, \tag{3.3}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are $m \times m$ matrices. The next generation matrix is then defined as $\boldsymbol{A}\boldsymbol{B}^{-1}$. The basic reproductive value of $\mathcal{R}_0$ is subsequently the spectral radius or largest eigenvalue of the next generation matrix. In the case of the SIR model, $\boldsymbol{F}(\boldsymbol{X}(t)) := -\beta SI/N$, while $\boldsymbol{V}(\boldsymbol{X}(t)) := \gamma I$. It follows that the necessary Jacobians evaluated at the disease-free equilibrium of $(N,0,0)$ results in $\boldsymbol{A}\boldsymbol{B}^{-1} = \beta/\gamma$ and agree with the previous section.

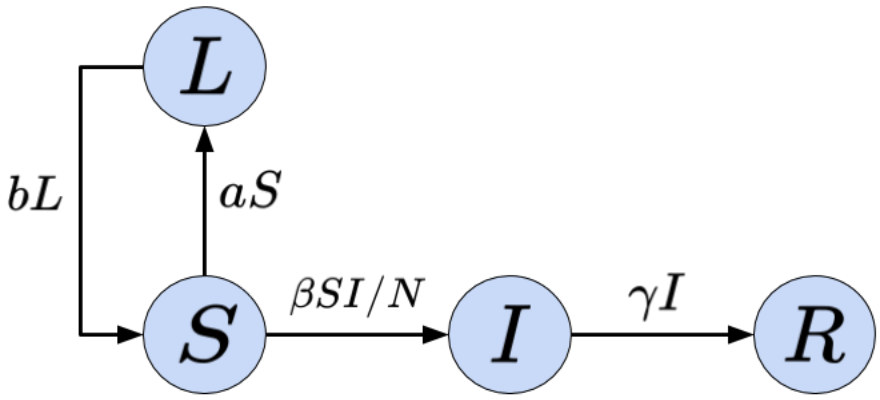


Figure 3.2: Proposed SLIR compartmental model capturing reduction in susceptible population

3.3 Methods

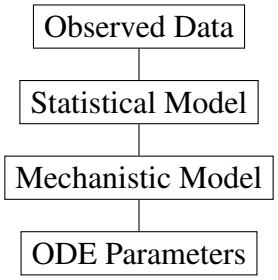
In this section, we detail the construction of our compartmental model designed to formulate an understanding of how reduction in human mobility might have altered early infection dynamics within New York City. The model is parsimonious, in that it consists of only four compartments and shares many well-established mathematical properties of the SIR model discussed in the background section. The dynamical system proposed comprises a closed population divided into susceptible, lockdown (L), infectious, and removed states. Since the time horizon under investigation is short, we choose to ignore demographic factors such as birth, death, and migration. Below in Figure 3.2 is a visualization of the population progression through the different states. The system of differential equations describing the changing system are

$$\begin{cases} \frac{dS(t)}{dt} &= -\beta S(t)I(t)/N - aS(t) + bL(t) \\ \frac{dL(t)}{dt} &= aS(t) - bL(t) \\ \frac{dI(t)}{dt} &= \beta S(t)I(t)/N - \gamma I(t) \\ \frac{dR(t)}{dt} &= \gamma I(t), \end{cases} \tag{3.4}$$

with initial conditions of $S(0) = N - i_0$, $L(0) = 0$, $I(0) = i_0$, and $R(0) = 0$. An important qualitative feature of our model is that susceptible individuals are temporarily moved into the lockdown compartment to reflect social distancing, mitigation measures, and reduced mobility. Over time, individuals are reintroduced into the susceptible population out of the lockdown state. Through an application of the next generation matrix method described in section 3.2, $\mathcal{R}_0$ can be seen to be equivalent to the standard SIR model, i.e. $\mathcal{R}_0 = \beta/\gamma$.

3.3.1 A Bayesian Hierarchical Model

We present our methodology in a general hierarchical framework to facilitate Bayesian inference of compartmental system parameters in equation (3.4). In this hierarchical formulation, we can be explicit about the role of the mechanistic system, requisite numerical integration, and underlying process parameters. This section will construct the statistical model piece-by-piece. The hierarchy of connected components in the model can be visualized bottom-up as follows,



We first establish notation to represent the mechanistic system of differential equations in the middle of the hierarchy. Let the equations in (3.4) be denoted by $\boldsymbol{F}$, where

$$\frac{d}{dt}\boldsymbol{X}(t) = \boldsymbol{F}(\boldsymbol{X}(t), t), \text{ where } \boldsymbol{X}(t) = \begin{pmatrix} S(t) \\ L(t) \\ I(t) \\ R(t) \end{pmatrix} \tag{3.5}$$

and

$$\boldsymbol{F}(\boldsymbol{X}(t),t) = \begin{pmatrix} -\gamma\mathcal{R}_0 S(t)I(t)/N - aS(t) + bL(t) \\ aS(t) - bL(t) \\ \gamma\mathcal{R}_0 S(t)I(t)/N - \gamma I(t) \\ \gamma I(t) \end{pmatrix} \tag{3.6}$$

and we have suppressed the dependence of $\{\mathcal{R}_0, \gamma, a, b\}$ in $\boldsymbol{F}(\boldsymbol{X}(t))$. The system is reparameterized in terms of $\mathcal{R}_0$ rather than β to be more epidemiologically interpretable and results from a simple transformation of $\beta = \gamma\mathcal{R}_0$. To recover the system states $\boldsymbol{X}(t)$, the system of differential equations must be solved. Given fixed values of $\mathcal{R}_0$, γ, a, and b, the solution to the system of differential equations is a vector-valued function

$$\boldsymbol{X}(t) = \int \boldsymbol{F}(\boldsymbol{X}(t),t;\mathcal{R}_0,\gamma,a,b)dt. \tag{3.7}$$

This solution will be necessary to connect with the observed data.

The top of the hierarchy is described by formulating a measurement process for the two outcome variables. Let the first outcome of interest be labeled $Y_L(t)$ and represent the percent of the population removed from the susceptible compartment by adhering to mitigation protocol. The subscript L is used for a reminder that this data is used to gain information on the lockdown compartment. Likewise, the second outcome is labeled $Y_I(t)$ and denotes the observed case counts over time. To model observation error in $Y_L(t)$, we choose a Beta distribution dependent upon a parameter ϕ_1 to control dispersion, i.e.,

$$Y_L(t)|L(t),\phi_1 \sim \text{Beta}\big(\phi_1 L(t)/N, \phi_1(1-L(t)/N)\big). \tag{3.8}$$

Notice $L(t)$ is necessarily scaled by the population size N to respect the support of the beta distribution. As we seek to inform the L compartment through cell phone mobility data, the beta distribution is a natural choice because the data will be anonymized and reported as a percentage of nominal

movement. This parameterization of the beta distribution has expectation and variance

$$\begin{aligned}\mathbb{E}[Y_L(t)|L(t),\phi_1] &= L(t)/N \\ \text{Var}(Y_L(t)|L(t),\phi_1) &= \frac{L(t)/N\big(1-L(t)/N\big)}{\phi_1+1}.\end{aligned}$$

To model observation noise in $Y_I(t)$, we use a negative binomial to account for overdispersion,

$$Y_I(t)|I(t),\phi_1 \sim \text{Negative Binomial}(I(t),\phi_2). \tag{3.9}$$

Stan provides an alternative parameterization of the negative binomial called `neg_binomial_2` with first two moments of

$$\begin{aligned}\mathbb{E}[Y_I(t)|I(t),\phi_2] &= I(t) \\ \text{Var}(Y_I(t)|I(t),\phi_2) &= I(t)+\frac{I(t)^2}{\phi_2}.\end{aligned}$$

In this way, ϕ_2 is viewed as a dispersion parameter. The full hierarchical description of the model can be completed by introducing prior distributions on the system parameters governing the differential

equations. Writing the model in full,

$$
\begin{aligned}
Y_L(t)|L(t),\phi_1 &\sim \text{Beta}\big(\phi_1 L(t)/N, \phi_1(1-L(t)/N)\big) \\
Y_I(t)|I(t),\phi_2 &\sim \text{Negative Binomial}(I(t),\phi_2) \\
\frac{d\boldsymbol{X}(t)}{dt} &= \boldsymbol{F}(\boldsymbol{X}(t),t;\mathcal{R}_0,\gamma,a,b) \\
\mathcal{R}_0|\gamma &\sim \text{log-normal}(0,1) \\
\gamma &\sim \text{Uniform}(0,1) \\
\phi_1 &\sim \text{Inverse Gamma}(0.1,0.1) \\
\phi_2 &\sim \text{Inverse Gamma}(0.1,0.1) \\
a &\sim \text{Beta}(1,5) \\
b &\sim \text{Uniform}(0,1).
\end{aligned}
\tag{3.10}
$$

The prior distributions on system parameters are weakly-informative. However, the prior distribution on a might at first appear suspect. Through prior predictive checks, we find that placing a uniform prior on a results in the SLIR model a priori favoring no epidemic breakout, as susceptibilities are removed from the population too quickly. Since the classical SIR model is a special case of our SLIR model as $a \to 0$, we place mass closer to 0 through the Beta(1,5) distribution to ensure the model generates reasonable predictions before seeing the data. The hierarchical model of the previous section crucially depends upon the numerical solution to a coupled set of differential equations of (3.4). In the appendix section, we detail the internal workings of Stan's numerical optimization routines. Finally, efficient Bayesian analysis of parameters within the set of nonlinear differential equations relies upon the efficiencies gained through Hamiltonian Monte Carlo (HMC). A detailed review of this methodology is also included in the appendix section.

3.4 Analysis and Results

In this section, we present two simulation studies and conclude with the New York City analysis. First, the proposed SLIR compartmental model is used to simulate data from two lockdown scenarios that affect human mobility differently. We then fit our Bayesian model to assess whether the true parameter values are adequately recovered. After, we analyze the real-world mobility and case count data that initially inspired the model formulation.

3.4.1 Simulated Data

To illustrate the nonlinear dynamics of which our model can capture, the first simulation reflects the idealized scenario of strict adherence to lockdown and mitigation measures, when population movement is quickly reduced in the early stages of the outbreak and remains reduced for the next 90 days. In this case, individuals move from the S compartment to the L compartment quickly and are slowly reintroduced into the susceptible population so that population movement decreases to about 60% relative to baseline within the first 20 days.

In the second scenario, we consider weak adherence to mitigation measures and illustrate the substantial change in dynamics by only altering the speed of flow back into the susceptible population from the L compartment. In this case, the peak percentage of the population in the lockdown compartment is 30% but quickly diminishes. In both simulations, the population size is fixed to $N = 10,000$, $i_0 = 1$, and the true data-generating process is shown below as a dashed red line. The median of the posterior predictive distribution and 95% credible intervals are shown in blue.

We fit our model to both scenarios using Stan's NUTS algorithm with 4 chains and 5,000 iterations each, the first half of which are discarded as warm-up. The convergence of the parameter chains are judged by inspecting the trace plots along with the Gelman-Rubin $\hat{R}$ values, which compares the variation between chains to the variation within Gelman and Rubin (1992). Ideally, the $\hat{R}$ value is close to one. These simulation results are displayed in Table 3.1.

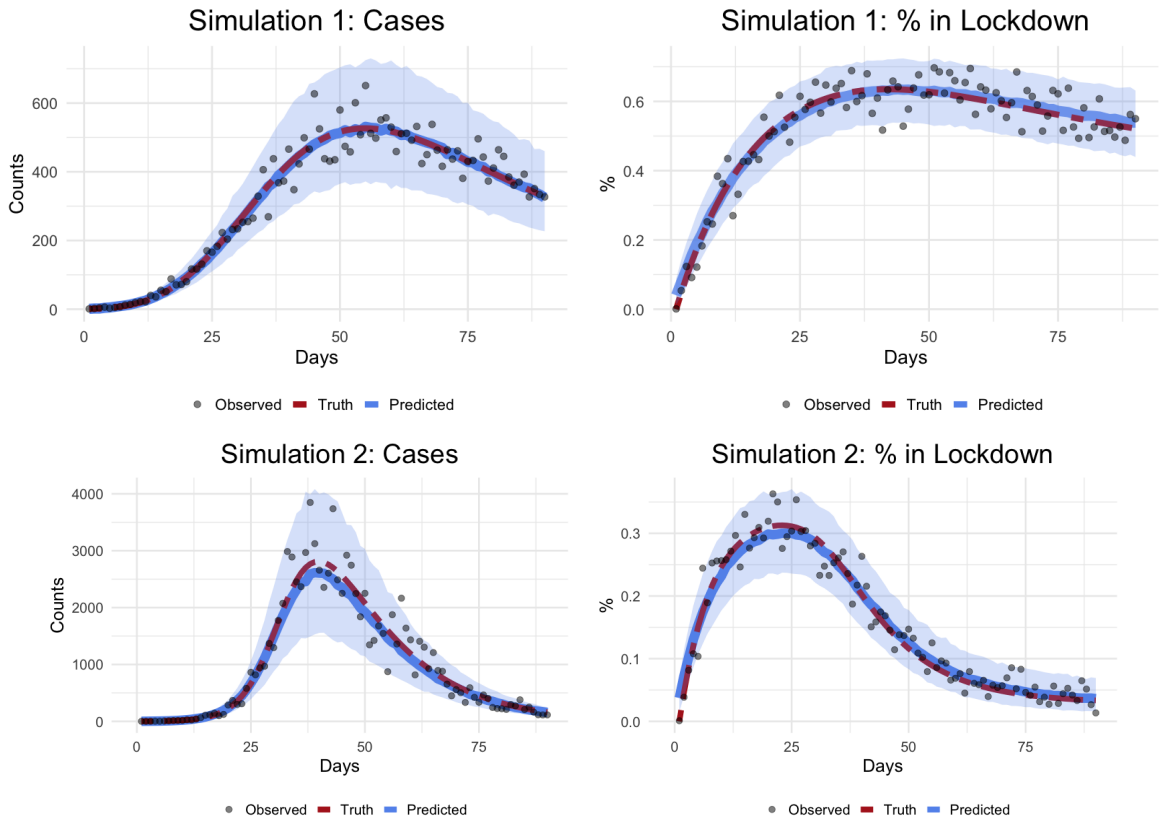


Figure 3.3: Model fit to simulated mobility and case count data

Table 3.1 Simulation Study

Data	Parameter	Truth	Median	95%-credible interval	G-R $\hat{R}$
Simulation 1	$\mathcal{R}_0$	5	4.93	4.720 - 5.130	1.00
	γ	0.1	0.090	0.086 - 0.150	1.00
	a	0.05	0.045	0.040 - 0.051	1.00
	b	0.1	0.095	0.040 - 0.051	1.00
Simulation 2	$\mathcal{R}_0$	5	4.79	4.590 - 5.010	1.00
	γ	0.1	0.099	0.096 - 0.104	1.00
	a	0.05	0.045	0.040 - 0.051	1.00
	b	0.1	0.095	0.086 - 0.106	1.00

In both cases, the Bayesian hierarchical model is able to infer the structural parameters of the SLIR model. Notice the change in case counts of both scenarios resulting from different susceptible population sizes.

3.4.2 New York City Analysis

The entire lead-up thus far was requisite background material for compartmental model inference and application to real-world data. As mentioned in the introduction, our main motivation for this article was to understand how a reduction in mobility affected early Covid-19 dynamics specifically within NYC. For convenience, we restate the data sources. Case counts are reported by the official website of the City of New York, available at `https://www1.nyc.gov/site/doh/covid/covid-19-data.page` and mobility data is hosted at `https://covid19.apple.com/mobility`. Since the Apple mobility data reflects a percent decrease in movement, it must first be transformed by subtraction from unity to adhere with the SLIR compartmental model structure. In other words, to prepare the mobility data for use in the hierarchical model, it must first be subtracted from one so that it no longer represents a percent decrease in transit mobility but rather a percent increase in individuals adhering to mitigation measures. Finally, we mention again that although the first case of Covid-19 in New York City was recorded on February 29th, we align our movement and case data to begin on March 8th, 2020, the day after Governor Andrew Cuomo declared a state of emergency in New York State. Finally, we take the initial number of cases to be the cases recorded on March 8th, 2020, and the population is fixed at 8,336,817 as determined by the US Census U.S. Census Bureau (2019).

To achieve parameter calibration, we fit in Stan our full hierarchical model using four chains run for 10,000 iterations each. We discard the first 5,000 as warm-up. Sufficient posterior exploration is assessed by examining parameter chain plots below and assessing $\hat{R}$ values, shown in Table 3.2.

The fitted time series are presented below on the right, along with the raw data used to train the model. On the left, prior predictive distributions are included to illustrate the degree in which

Table 3.2 New York City Analysis

Data	Parameter	Median	95%-credible interval	G-R $\hat{R}$
New York City	$\mathcal{R}_0$	5.130	4.841 - 5.448	1.00
	γ	0.212	0.191 - 0.234	1.00
	a	0.115	0.106 - 0.124	1.00
	b	0.022	0.019 - 0.024	1.00

Bayesian learning occurs after observing the data. We also include the fit of an SIR model to illustrate its inability to capture the dynamics.

To assess the structural fit of our hypothesized SLIR mechanism, we next interpret parameter values to ensure they are logical and consistent with outside literature. The $\mathcal{R}_0$ estimate is cross-referenced with those of other popular online models. Using only death statistics as reported by Johns Hopkins University, Gu (2020) embeds a SEIR model in a machine learning framework for many regions across the United States and 70 countries. In this work, $\mathcal{R}_0$ is estimated for NYC to be between 5.0 and 5.8, in close agreement with our model. As an additional point of reference, Ives and Bozzuto (2020) provide an alternative methodology that fits a time series state-space model to death counts and explicitly accounts for reporting delays. $\mathcal{R}_0$ is estimated to be 6.3 with a 95% confidence interval of 4.5-9. Our model thus has the added benefit of mechanistic interpretability as well as a tighter credible interval.

The posterior of γ has a median value of 0.21 and a 95% credible interval of (0.191, 0.234). From this, we arrive at an estimate of approximately 5 days for the average infectious removal time. This estimate could be reasonable assuming asymptomatic or presymptomatic transmission is possible and that individuals isolate at the onset of symptoms; see Gandhi et al. (2020) for evidence. Outside work estimates symptom onset time to also be around 5 days. For example, Lauer et al. (2020) use 181 confirmed Covid-19 cases to estimate a median symptom onset time of 5.1 days with a 95% confidence interval (4.5, 5.8) days. A systematic review by Madjid et al. (2020) estimates a mean symptom onset time of 5.2 days with a 95% confidence interval of (4.1, 7.0) days. These estimates

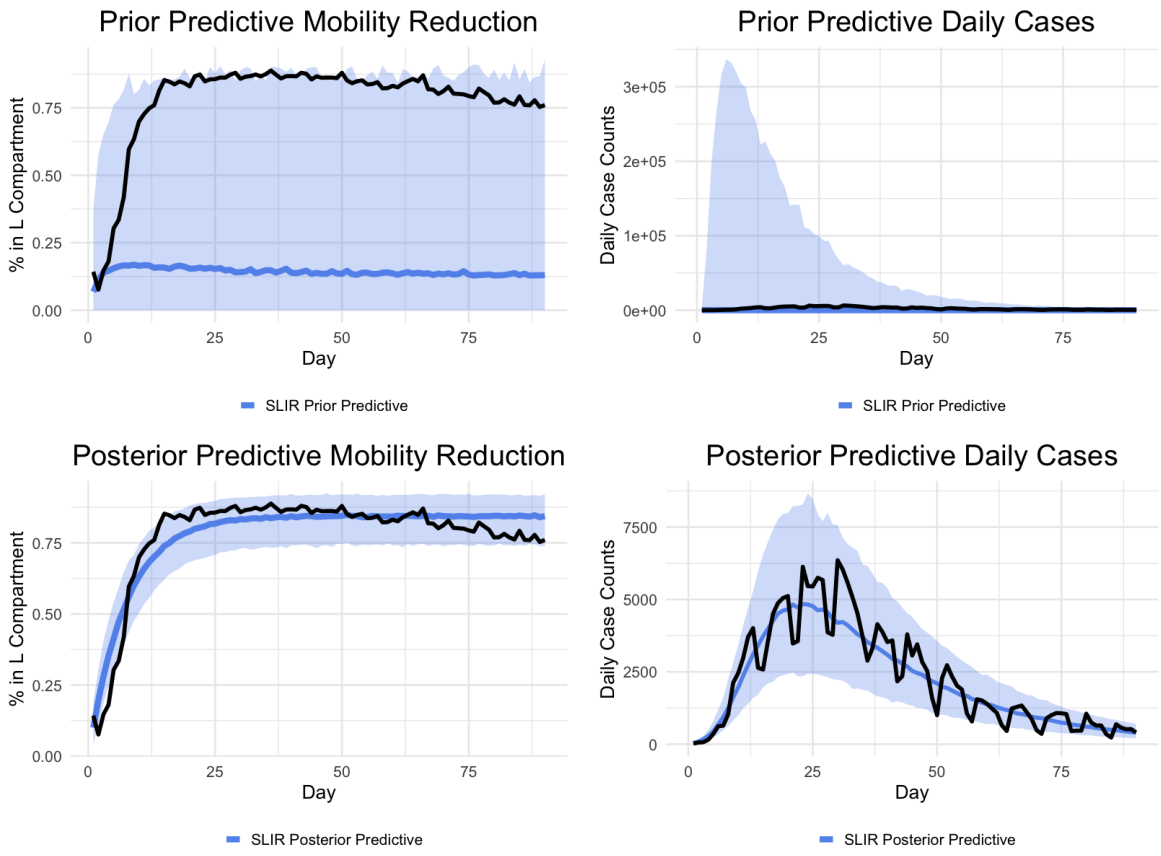


Figure 3.4: SLIR Prior and Posterior Predictive Checks

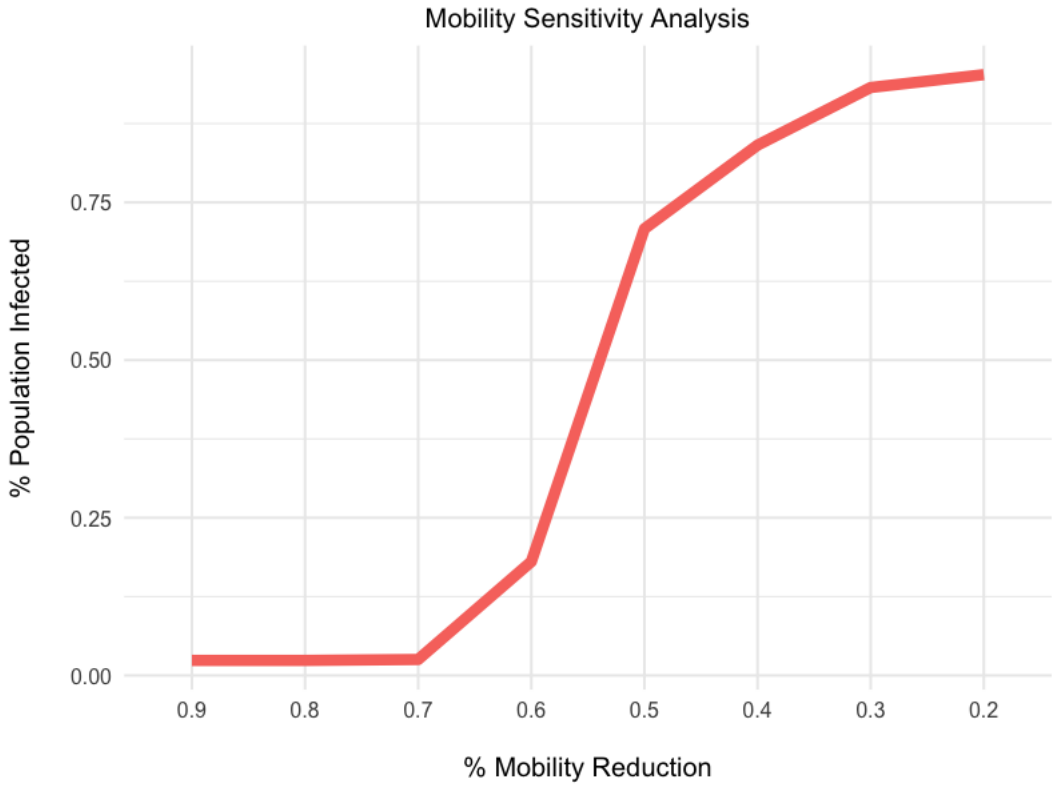


Figure 3.5: Counterfactual scenarios demonstrating change in susceptible population levels

provide credence in establishing a correspondence between population movement reduction and infection dynamics with the mechanistic SLIR model.

3.4.3 Sensitivity Analysis and Out-of-sample Forecasting

We conclude the New York City analysis by assessing the sensitivity of the model to changing mobility levels. Additionally, we assess out-of-sample predictive capacity of the SLIR model. To perform the sensitivity analysis, we first generate a range of hypothetical mobility scenarios, from full mobility reduction through extremely stringent lockdown measures to a more mild decline. We then conduct the same forecasting analysis by training only upon limited observed data, in this case a three-week observation period. As we vary the size of the susceptible population, Figure 3.5 displays changing case counts.

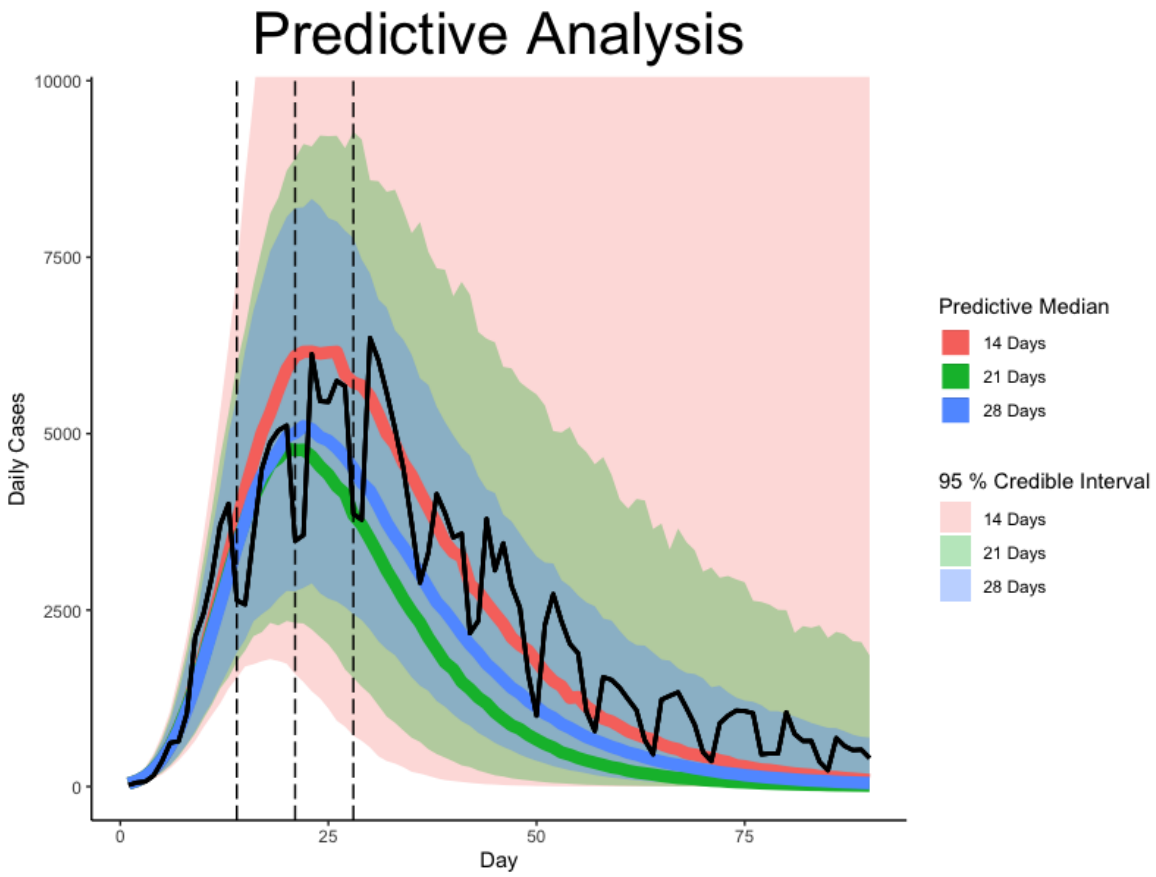


Figure 3.6: SLIR Out-of-sample Forecasts

It is important to note the explosive case growth with mobility reduction levels under 60% due to the nonlinear dynamics present in SIR-type models. This is evidence that a mobility reduction significantly altered infections within the city, assuming SIR-type dynamics.

To conclude this section, we analyze the out-of-sample forecasting ability of the SLIR model when trained on only a subset of the ninety-day period. We consider for illustration a two, three, and four-week training intervals. These are indicated by vertical dashed lines in Figure 3.6. The predictive median is illustrated with a solid line. With only two weeks observed, the predictive median is reasonably representative of the future trajectory, but with very large uncertainty. The upper 95% predictive curve for the two-week window reaches roughly 400,000 infections, but is clear that the predictive interval quickly contracts as more data is observed (cf. Figure 3.4). Both the three week

and four-week training periods have contracted credible interval forecasts that contain the observed data. Finally, we mention that we were unable to fit the standard SIR model to the New York City data. Using the quasi-Newton optimization functions available in Stan, we found the SIR model fit was highly unstable across a range of starting values, indicating extreme multi-modality in the likelihood surface.

3.5 Discussion and Future Work

In this work, we have formulated a basic extension of the classical SIR model to jointly fit cell phone transit mobility data and case counts. By jointly modeling two outcome variables, we establish a mechanistic correspondence between reduced mobility and infection dynamics. The applied analysis and findings, however, are limited to NYC during the first 90 days. The disease progression throughout the region was well-approximated by deterministic dynamics and modeling with a simple compartment system. In other geographic regions, the dynamics might not be as suitably well-behaved or understood. It is difficult to capture the myriad of factors contributing to disease spread throughout a population, and often a stochastic model may be more appropriate. Additionally, the time horizon considered in our applied analysis is relatively short. As a disease progresses throughout the population and becomes more widespread, the strategy of informing the susceptible population numbers through cell phone mobility data becomes more difficult. It is also worth consideration about whether, in general, smartphone mobility data can be considered a representative sample of individuals' adherence to mitigation protocols. It is therefore difficult to build a quantitative understanding of the dynamics between lockdown measures and disease progression over long periods of time. A direction for future research is to establish modeling strategies for such scenarios. This will introduce additional challenges of accounting for population immunity, vaccinations, and natural seasonality effects.

3.5.1 Stochastic Model Extensions

We briefly discuss for both discrete and continuous time how the deterministic SLIR model can be relaxed through a state-space (or Hidden Markov Model) framework, where stochasticity is introduced into the underlying driving epidemic process. We first discuss a discrete-time extension by modifying equation (3.10) so that the numerical solution to $\frac{d\boldsymbol{X}(t)}{dt}$ is embedded within a likelihood function at the discrete observation time points. In this way, equation (3.10) is modified to include $p(\boldsymbol{X}(t)|\boldsymbol{\theta},\mathcal{R}_0,\gamma,a,b)$, where $\boldsymbol{\theta}$ is introduced to accommodate new parameters. In other words, stochasticity is introduced to the driving dynamics through the choice of a suitable probability distribution that is a function of the numerical solution of the system of differential equations. This approach has been successfully applied to forecast seasonal influenza Osthus et al. (2017), Dukic et al. (2012), as well as to assess Covid-19 interventions in China Wang et al. (2020b) and Japan Kobayashi et al. (2020).

An alternative approach is to incorporate continuous-time stochasticity directly by expressing $d\boldsymbol{X}(t)$ as a stochastic differential equation (SDEs) Øksendal (2003). In this case, one modification of equation (3.10) could be

$$d\boldsymbol{X}(t) = \boldsymbol{\mu}(\boldsymbol{X}(t),t)dt + \boldsymbol{L}(\boldsymbol{X}(t),t)d\boldsymbol{W}(t), \tag{3.11}$$

where $\boldsymbol{W}(t) = (W_1(t),W_2(t),W_3(t),W_4(t))^\top$ is a vector of standard Wiener processes or Brownian motions, $\boldsymbol{\mu}$ is some vector-valued function, and $\boldsymbol{L}$ is a compatible matrix. Depending on the structure of $\boldsymbol{\mu}$, there may or may not exist an explicit solution. A SDE that affords a closed form solution is the Ornstein-Uhlenbeck (OU) process. See Chatzilena et al. (2019) for Stan code in the case where the infectious compartment of the SIR model is endowed an OU process to allow for random fluctuations. See also Wang et al. (2018) for an analysis of the susceptible-infectious-susceptible model where time-varying parameters are introduced through an OU process. In both examples, the analytic nature of the solution to the SDE enables efficient implementation in Stan or alternative software. In cases where no closed-form solution to (3.11) exists, inference in such a system is closely related

to Bayesian filtering and smoothing, amenable to such methods as the Kalman filter and extensions Särkkä and Solin (2019). The New York City data were captured well by deterministic dynamics, but often elsewhere disease progression is not suitably captured within such a framework. Bayesian inference for SDEs modeling infectious disease dynamics is thus an attractive area of future research.

3.6 Appendix

3.6.1 Numerical Solvers

The hierarchical model of the previous section crucially depends upon the numerical solution to a coupled set of differential equations. We briefly review the popular numerical integration schemes and connect them with our hierarchical model of the compartmental system in (3.4). In practice, solutions to the SIR model are numerically approximated and entail specific computational challenges. Runge-Kutta (RK) is a classical numerical method Struthers and Potter (2019); Press et al. (2007) in which we employ to numerically solve this set of nonlinear differential equations. RK generalizes the well-known Euler method for iteratively solving systems of differential equations. In the following, we formulate the RK method in vector notation for the proposed SLIR compartmental model.

Consider $\boldsymbol{X}(t)$ in (3.5). From the fundamental theorem of Calculus, we know that

$$\boldsymbol{X}(t+\Delta t) = \boldsymbol{X}(t) + \int_t^{t+\Delta t} \boldsymbol{F}(\boldsymbol{X}(u), u) du \,. \tag{3.12}$$

Given the initial value $\boldsymbol{X}(t_0)$ at time t_0, numerically solving for $\boldsymbol{X}(t)$ amounts to constructing a sequence $\{\boldsymbol{X}_n : n = 0, 1, \ldots, T\}$, where $\boldsymbol{X}_n := \boldsymbol{X}(t_n)$ and $t_n = t_{n-1} + \Delta t$ are equispaced time points for $n = 0, 1, \ldots, T$, by approximating the integral of $\boldsymbol{F}(\boldsymbol{X}(t), t)$ in (3.12). Using this approximation, a sequence is generated starting from some t_0 as,

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \int_{t_n}^{t_n+\Delta t} \boldsymbol{F}(\boldsymbol{X}(u), u) du \,. \tag{3.13}$$

where the initial condition $\boldsymbol{X}_0 := \boldsymbol{X}(t_0)$ is given. There are several customary choices for such approximations, but for most practical purposes one need not look beyond the following:

$$\int_t^{t+\Delta t} \boldsymbol{F}(\boldsymbol{X}(t),t)dt = \begin{cases} (\Delta t)\boldsymbol{F}(\boldsymbol{X}(t),t) \\ \frac{(\Delta t)}{2}\left(\boldsymbol{F}(\boldsymbol{X}(t),t) + \boldsymbol{F}(\boldsymbol{X}(t+\Delta t),t+\Delta t)\right) \\ (\Delta t)\boldsymbol{F}(\boldsymbol{X}(t+(\Delta t)/2),t+(\Delta t)/2)\ . \end{cases} \tag{3.14}$$

Euler's approximation (first equation in (3.14)) is the simplest of the approximations which yields Euler's Method. In this case, the sequence is constructed for $n = 0, 1, \ldots, T-1$ as

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + (\Delta t)\boldsymbol{F}(\boldsymbol{X}_n, t_n)\ . \tag{3.15}$$

Starting with $\boldsymbol{X}_0$, each element of the sequence in (3.15) is computed since $\boldsymbol{F}(\boldsymbol{X}_n, t_n)$ is available. The spacing between the time points, Δt, is specified by the user and controls the resolution of the numerical solution.

Euler's approximation is easy to execute and based upon a first order Taylor expansion. It is also the least accurate. The Trapezoidal and Modified Euler approximations in (3.14) are based upon numerical integration using trapezoidal areas and the midpoint approximation. Both these methods help improve Euler's method with the Modified Euler outperforming the trapezoidal rule in terms of accuracy. However, the Trapezoidal and the Modified Euler methods involve $\boldsymbol{X}_{n+1}$ and $\boldsymbol{X}_{n+1/2} := \boldsymbol{X}(t_{n+1/2})$ with $t_{n+1/2} := t_n + (\Delta)t/2$, respectively, which are unknown at iteration n. In fact, $\boldsymbol{X}_{n+1}$ is the very quantity we wish to compute at iteration n. Therefore, we substitute these unknown quantities with their first order (Euler) approximations in (3.15) that are available at iteration n. For each $n = 0, 1, \ldots, T-1$, we compute

$$\begin{aligned} \boldsymbol{a}_n &= \boldsymbol{F}\left(\boldsymbol{X}_n, t_n\right) \\ \boldsymbol{b}_n &= \boldsymbol{F}\left(\boldsymbol{X}_n + (\Delta t)\boldsymbol{a}_n, t_{n+1}\right) \\ \boldsymbol{c}_n &= \boldsymbol{F}\left(\boldsymbol{X}_n + (\Delta t/2)\boldsymbol{a}_n, t_{n+1/2}\right) \end{aligned} \tag{3.16}$$

and the transition from $\boldsymbol{X}_n$ to $\boldsymbol{X}_{n+1}$ as

$$\boldsymbol{X}_{n+1} = \begin{cases} \boldsymbol{X}_n + (\Delta t)\frac{(\boldsymbol{a}_n + \boldsymbol{b}_n)}{2} & \text{(Trapezoidal)}\ ; \\ \boldsymbol{X}_n + (\Delta t)\boldsymbol{c}_n & \text{(Modified Euler)} \end{cases} \tag{3.17}$$

Both methods in (3.17) deliver noticeable improvements over Euler's method in (3.15). Numerical error from (3.17) are much smaller in magnitude and grow less quickly. This can be explained by observing that while (3.15) depends only upon the data available at t_n (only one data point), the two methods in (3.17) use current data at t_n along with estimates of the slope at a point that lies in the future. While these estimates are computed using only the currently available data, they still produce substantially improved estimates. Also see Treiber and Kanagaraj (2015) for comparisons among different numerical integration schemes.

Higher order Taylor expansions produce other iterative schemes. Thus, second order methods emerge from

$$\begin{aligned} \boldsymbol{X}(t+\Delta t) &= \boldsymbol{X}(t) + (\Delta t)\boldsymbol{F}(\boldsymbol{X}(t),t) \\ &\quad + \frac{(\Delta t)^2}{2}\frac{d}{dt}\boldsymbol{F}(\boldsymbol{X}(t),t) + O((\Delta t)^2)\ . \end{aligned} \tag{3.18}$$

A second order iterative scheme corresponding to (3.18) updates

$$\begin{aligned} \boldsymbol{X}_{n+1} &= \boldsymbol{X}_n + (\Delta t)\boldsymbol{F}(\boldsymbol{X}_n,t_n) \\ &\quad + \frac{(\Delta t)^2}{2}\left\{ \left.\frac{\partial \boldsymbol{F}}{\partial t}\right|_{t=t_n} + \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right]\Bigg|_{\boldsymbol{X}=\boldsymbol{X}_n} \left.\frac{d}{dt}\boldsymbol{X}(t)\right|_{t=t_n} \right\}\ , \end{aligned} \tag{3.19}$$

where we have used the multivariable chain-rule of derivatives to evaluate the derivative of $\boldsymbol{F}(\boldsymbol{X}(t),t)$ with respect to t, $\left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right]$ is the matrix with (i,j)-th element being the partial derivative of the i-th element of $\boldsymbol{F}(\boldsymbol{X}(t),t)$ with respect to the j-th variable in $\boldsymbol{X}$, and $\left.\frac{d}{dt}\boldsymbol{X}(t)\right|_{t=t_n} = \boldsymbol{F}(\boldsymbol{X}_n,t_n)$. Unfortunately, computing (3.19) requires the derivatives of $\boldsymbol{F}(\boldsymbol{X}(t),t)$ and may, in general, be numerically cumbersome.

The Runge-Kutta methods are among the most conspicuous of numerical methods for solving systems of ordinary differential equations. The underlying idea is to achieve the same accuracy as Taylor series updates without requiring higher order derivatives of $\boldsymbol{F}(\boldsymbol{X}(t))$. We can motivate this approach from the earlier methods. In (3.17), the Trapezoidal and Modified Euler methods define the updates using $\boldsymbol{a}_n$, $\boldsymbol{b}_n$ and $\boldsymbol{c}_n$ that are completely specified. In particular, observe that the Trapezoidal method updates using a weighted average of $\boldsymbol{a}_n$ and $\boldsymbol{b}_n$. Instead of prescribing $\boldsymbol{a}_n$ and $\boldsymbol{b}_n$, the Runge-Kutta approach prefers to find weighted averages to ensure that the approximation matches that from a Taylor series expansion such as in (3.18) or (3.19). Therefore, a second-order Runge-Kutta method (RK2) writes

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + (\Delta t)\left\{\omega_1 \boldsymbol{a}_n + \omega_2 \boldsymbol{F}(\boldsymbol{X}_n + (\Delta t)\beta \boldsymbol{a}_n, t_n + (\Delta t)\alpha)\right\} \tag{3.20}$$

and seeks to find ω_1, ω_2, α and β so that the approximation matches (3.19). Substituting the first-order expansion,

$$\begin{aligned}
&\boldsymbol{F}(\boldsymbol{X}_n + (\Delta t)\beta \boldsymbol{a}_n, t_n + \alpha(\Delta t)) = \boldsymbol{F}(\boldsymbol{X}_n, t_n) \\
&\quad + (\Delta t)\beta \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right]\Big|_{\boldsymbol{X}=\boldsymbol{X}_n} \boldsymbol{a}_n + (\Delta t)\alpha \frac{\partial \boldsymbol{F}}{\partial t}\Big|_{t=t_n} + O((\Delta t)^2) \\
&= \boldsymbol{a}_n + (\Delta t)\beta \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right]\Big|_{\boldsymbol{X}=\boldsymbol{X}_n} \boldsymbol{a}_n \\
&\quad + (\Delta t)\alpha \left\{ \frac{\partial \boldsymbol{F}}{\partial t}\Big|_{t=t_n} + \left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{X}}\right] \frac{d}{dt}\boldsymbol{X}(t)\Big|_{t=t_n} \right\} + O((\Delta t)^2) \,,
\end{aligned} \tag{3.21}$$

into the right hand side of (3.20) and comparing with the expansion (3.19) we find that the two expansions are equivalent if

$$\omega_1 + \omega_2 = 1 \,; \quad \omega_2 \beta = \omega_2 \alpha = 1/2 \,. \tag{3.22}$$

RK2 specifies $\omega_1 = \omega_2 = 1/2$ and $\alpha = \beta = 1$, which, when substituted into (3.20), yields the Trapezoidal approximation.

More generally, the explicit RK methods of order s specify updating schemes

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + (\Delta t) \sum_{i=1}^{s} \omega_i \boldsymbol{k}_i \,, \tag{3.23}$$

where

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{F}(\boldsymbol{X}_n, t_n + (\Delta t)\alpha_1) \\
\boldsymbol{k}_i &= \boldsymbol{F}\left(\boldsymbol{X}_n + (\Delta t) \sum_{j=1}^{i-1} \beta_{ij} \boldsymbol{k}_j, t_n + (\Delta t)\alpha_i\right) i = 2, \ldots, s.
\end{aligned}$$

The coefficients are found from an s-th order Taylor expansion. A popular choice sets $\alpha_1 = 0$ and solves

$$\sum_{i=1}^{s} \omega_i = 1 \;\text{ and }\; \sum_{j=1}^{i-1} \beta_{ij} = \alpha_i \;\text{ for }\; i = 2, 3, \ldots, s \,. \tag{3.24}$$

In particular, the RK4 method specifies the following values after taking $s = 4$: (3.23):

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix}, \tag{3.25}$$
$$\begin{bmatrix} \beta_{21} & & \\ \beta_{31} & \beta_{32} & \\ \beta_{41} & \beta_{42} & \beta_{43} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & & \\ 0 & \frac{1}{2} & \\ 0 & 0 & 1 \end{bmatrix}.$$

The appropriate step size Δt is hard to determine. An advantage of Stan's implementation is an adaptive step size is used by comparing the solution obtained using the four term approximation of above as well as a five term approximation. If these approximations agree, the algorithm proceeds, otherwise a new step size is calculated. More details can be found in the user manual Stan Development Team (2020), but the result is a fast, efficient procedure of high accuracy.

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

Our choice of implementation in Stan, as opposed to more traditional BUGS or JAGS Plummer (2003), is pragmatic. First, the latter languages are declarative and built upon graphical models. In contrast, Stan is a fully imperative programming language. Additionally, built-in differential equation routines are included such as the Runge-Kutta numerical solver described in the previous section. This makes the software implementation of our model more natural and readable. More importantly, however, the parameters in a nonlinear compartmental model are often highly correlated, as demonstrated in below in Figure 3.7.

Hamiltonian Monte Carlo (HMC) is more equipped to sample from complex posterior distributions with high autocorrelations than standard Metropolis schemes. The most popular presentation of Hamiltonian Monte Carlo is by way of analogy with statistical mechanics. Let $\boldsymbol{\theta}$ be an arbitrary d-dimensional parameter vector. To sample efficiently from the posterior of $\boldsymbol{\theta}$ after conditioning on data, an idealized physical system is introduced to leverage the geometry of the underlying manifold on which $\boldsymbol{\theta}$ lives. We will not embark upon a comprehensive development of HMC here, referring the reader to excellent introductory expository articles by

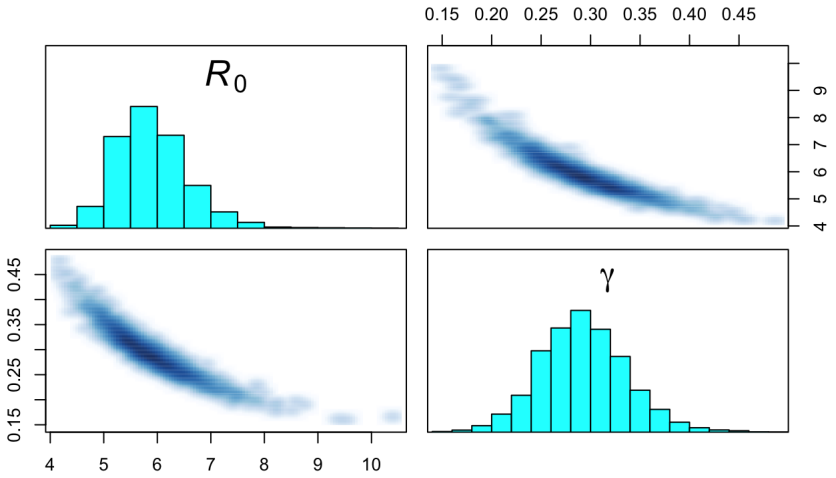


Figure 3.7: $\mathcal{R}_0$ and γ joint posterior

Neal (2012) and Betancourt (2018). Instead, we provide a heuristic account of the HMC algorithm and why it works.

We begin by recalling the more conspicuous Metropolis random walk and the concept of detailed balance. Let $p(\boldsymbol{\theta}|\boldsymbol{Y})$ be the posterior distribution from which we wish to sample. As it may be difficult to directly sample from $p(\boldsymbol{\theta}|\boldsymbol{Y})$, the Metropolis random-walk algorithm constructs a Markov chain with $p(\boldsymbol{\theta}|\boldsymbol{Y})$ as its stationary distribution. Given an initial value $\boldsymbol{\theta}^{(0)}$, at iteration t we draw a "proposed" value $\boldsymbol{\theta}^*$ from a symmetric distribution $q(\cdot|\boldsymbol{\theta}^{(t-1)})$. A simple yet effective choice for many applications is $q(\cdot|\boldsymbol{\theta}^{(t-1)}) = N(\cdot|\boldsymbol{\theta}^{(t-1)}, \boldsymbol{V})$, where $\boldsymbol{V}$ is a fixed variance covariance matrix that helps tune the algorithm, although in general the proposal can be generated from any symmetric distribution. After generating $\boldsymbol{\theta}^*$ we simulate a coin with probability of heads $\min\left(1, \frac{p(\boldsymbol{\theta}^*|\boldsymbol{Y})}{p(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{Y})}\right)$ and set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^*$ if it is a head. Otherwise we set $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t-1)}$. That $p(\boldsymbol{\theta}|\boldsymbol{Y})$ is indeed the desired stationary distribution can be seen as follows. Let us assume that the current state $\boldsymbol{\theta}^{(t-1)} = \boldsymbol{a}$ is a draw from $p(\boldsymbol{\theta}|\boldsymbol{Y})$ and consider the possibility of moving to $\boldsymbol{\theta}^{(t)} = \boldsymbol{b}$. This conditional probability is given

by the transition probability that a value of $\boldsymbol{b}$ is proposed from $q(\cdot|\boldsymbol{a})$ and that this value is accepted. Hence,

$$\begin{aligned} T(\boldsymbol{a} \to \boldsymbol{b}) &= P(\boldsymbol{\theta}^{(t)} = \boldsymbol{b} | \boldsymbol{\theta}^{(t-1)} = \boldsymbol{a}) \\ &= q(\boldsymbol{b}|\boldsymbol{a}) \min\left(1, \frac{p(\boldsymbol{b}|\boldsymbol{Y})}{p(\boldsymbol{a}|\boldsymbol{Y})}\right) \\ &= \min\left(q(\boldsymbol{b}|\boldsymbol{a}), q(\boldsymbol{b}|\boldsymbol{a})\frac{p(\boldsymbol{b}|\boldsymbol{Y})}{p(\boldsymbol{a}|\boldsymbol{Y})}\right) . \end{aligned} \tag{3.26}$$

The form of the transition probability in (3.26) implies time-reversibility (or detailed balance) in the following sense:

$$\begin{aligned} P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{a}, \boldsymbol{\theta}^{(t)} = \boldsymbol{b}) &= P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{a}) T(\boldsymbol{a} \to \boldsymbol{b}) \\ &= p(\boldsymbol{a}|\boldsymbol{Y}) \min\left(q(\boldsymbol{b}|\boldsymbol{a}), q(\boldsymbol{b}|\boldsymbol{a})\frac{p(\boldsymbol{b}|\boldsymbol{Y})}{p(\boldsymbol{a}|\boldsymbol{Y})}\right) \\ &= \min\left(p(\boldsymbol{a}|\boldsymbol{Y})q(\boldsymbol{b}|\boldsymbol{a}), q(\boldsymbol{b}|\boldsymbol{a})p(\boldsymbol{b}|\boldsymbol{Y})\right) \\ &= \min\left(p(\boldsymbol{a}|\boldsymbol{Y})q(\boldsymbol{a}|\boldsymbol{b}), q(\boldsymbol{a}|\boldsymbol{b})p(\boldsymbol{b}|\boldsymbol{Y})\right) \\ &= p(\boldsymbol{b}|\boldsymbol{Y})q(\boldsymbol{a}|\boldsymbol{b}) \min\left(\frac{p(\boldsymbol{a}|\boldsymbol{Y})}{p(\boldsymbol{b}|\boldsymbol{Y})}, 1\right) \\ &= P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{b}) T(\boldsymbol{b} \to \boldsymbol{a}) \\ &= P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{b}, \boldsymbol{\theta}^{(t)} = \boldsymbol{a}) , \end{aligned} \tag{3.27}$$

where we have used the symmetry $q(\boldsymbol{a}|\boldsymbol{b}) = q(\boldsymbol{b}|\boldsymbol{a})$ in the fourth equality in (3.27). It follows that the draw of $\boldsymbol{\theta}^{(t)}$ is also from $p(\boldsymbol{\theta}|\boldsymbol{Y})$ because

$$\begin{aligned} P(\boldsymbol{\theta}^{(t)} = \boldsymbol{b}) &= \int P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{a}, \boldsymbol{\theta}^{(t)} = \boldsymbol{b}) d\boldsymbol{a} \\ &= \int P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{b}, \boldsymbol{\theta}^{(t)} = \boldsymbol{a}) d\boldsymbol{a} \\ &= P(\boldsymbol{\theta}^{(t-1)} = \boldsymbol{b}) \\ &= p(\boldsymbol{b}|\boldsymbol{Y}) . \end{aligned} \tag{3.28}$$

The underlying idea behind HMC is that instead of generating the proposed value from a random distribution, we use a deterministic **symplectic integrator** to propose $\boldsymbol{\theta}^*$. This symplectic integrator is designed based upon Hamiltonian dynamics. Suppose that we wish to sample from $p(\boldsymbol{\theta}|\boldsymbol{Y})$, where $\boldsymbol{\theta} \in \mathbb{R}^d$. We in-

troduce an auxiliary variable $\boldsymbol{r} \in \mathbb{R}^d$ so that we can efficiently sample from the joint density $p(\boldsymbol{\theta}, \boldsymbol{r}|\boldsymbol{Y})$. If $(\boldsymbol{\theta}^{(t)}, \boldsymbol{r}^{(t)}) \sim p(\boldsymbol{\theta}, \boldsymbol{r}|\boldsymbol{Y})$, then

$$\begin{aligned} P(\boldsymbol{\theta}^{(t)} = \boldsymbol{b}) &= \int P(\boldsymbol{\theta}^{(t)} = \boldsymbol{b}, \boldsymbol{r}^{(t)} = \boldsymbol{u}) d\boldsymbol{u} \\ &= \int p(\boldsymbol{b}, \boldsymbol{u}|\boldsymbol{Y}) d\boldsymbol{u} \\ &= p(\boldsymbol{b}|\boldsymbol{Y}) \, . \end{aligned} \tag{3.29}$$

Hence, sampling from the joint density $p(\boldsymbol{\theta}, \boldsymbol{r}|\boldsymbol{Y})$ results in samples from $p(\boldsymbol{\theta}|\boldsymbol{Y})$.

The auxiliary variable, $\boldsymbol{r}$, is also called the "momentum" in Hamilton dynamics. For our purposes, it suffices to specify that $p(\boldsymbol{\theta}, \boldsymbol{r}|\boldsymbol{Y}) = p(\boldsymbol{\theta}|\boldsymbol{Y}) \times p(\boldsymbol{r})$. Therefore, $p(\boldsymbol{r}|\boldsymbol{\theta}, \boldsymbol{Y}) = p(\boldsymbol{r})$ which means that $\boldsymbol{r}$ is independent of the data $\boldsymbol{Y}$ and the model parameters $\boldsymbol{\theta}$. More specifically, we assume that $p(\boldsymbol{r}) = N(\boldsymbol{r}|\boldsymbol{0}, \boldsymbol{I}_d) \propto \exp\left(-\frac{1}{2}\boldsymbol{r}^\top \boldsymbol{r}\right)$. Therefore,

$$\log p(\boldsymbol{\theta}, \boldsymbol{r}|\boldsymbol{Y}) = \text{constant} + \log p(\boldsymbol{\theta}|\boldsymbol{Y}) - \frac{1}{2}\boldsymbol{r}^\top \boldsymbol{r} \, . \tag{3.30}$$

The above density can be looked upon as a physical system subject to Hamiltonian dynamics, where $\boldsymbol{\theta}$ is a particle's position in $\mathbb{R}^d$ and $\boldsymbol{r}$ is the particle's momentum.

In order to sample from (3.30), a simple HMC algorithm proceeds closely on the lines of the Metropolis random walk described earlier, but replaces the random generation of a proposed value for $\boldsymbol{\theta}$ by a symplectic integrator constructed from Hamiltonian dynamics. With the current state $(\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^{(t-1)})$, we begin iteration t by drawing the momentum variable $\boldsymbol{r}^* \sim N(\boldsymbol{0}, \boldsymbol{I}_d)$. Setting $\boldsymbol{r}^{(t)} = \boldsymbol{r}^*$ we perform L steps of a symplectic integrator (also known as "leapfrog"), where each step comprises the following:

$$\begin{aligned} \boldsymbol{r}^{(t+\varepsilon/2)} &= \boldsymbol{r}^{(t)} + \frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}); \\ \boldsymbol{\theta}^{(t-1+\varepsilon)} &= \boldsymbol{\theta}^{(t-1)} + \varepsilon \boldsymbol{r}^{(t+\varepsilon/2)}; \\ \boldsymbol{r}^{(t+\varepsilon)} &= \boldsymbol{r}^{(t+\varepsilon/2)} + \frac{\varepsilon}{2}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \, , \end{aligned} \tag{3.31}$$

where $\mathcal{L}(\boldsymbol{\theta}) = \log p(\boldsymbol{\theta}|\boldsymbol{Y})$. Let $\tilde{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{r}}$ be the output of (3.31) at the end of L steps. The values of $\tilde{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{r}}$ are considered the "proposed" values at iteration t and accepted as $(\boldsymbol{\theta}^{(t)}, \boldsymbol{r}^{(t)}) = (\tilde{\boldsymbol{\theta}}, -\tilde{\boldsymbol{r}})$ with acceptance probability $\min\left(1, \frac{p(\tilde{\boldsymbol{\theta}}|\boldsymbol{Y})p(\tilde{\boldsymbol{r}})}{p(\boldsymbol{\theta}^{(t-1)}|\boldsymbol{Y})p(\boldsymbol{r}^*)}\right)$. This last Metropolis step together with the negation of the momentum variable in the final update ensures time-reversibility as in (3.27) and, as seen in (3.28), maintains $p(\boldsymbol{\theta}|\boldsymbol{Y})$ as

the stationary distribution.

We provide some further intuition on the time-reversibility of the simple HMC algorithm. The key to this result is that the leapfrog iteration in (3.31) preserves volumes. To be slightly more precise, let $\mathcal{D}$ be a small region in the $(\boldsymbol{\theta}, \boldsymbol{r})$ space and suppose the L leapfrog steps maps $\mathcal{D}$ to a region $\tilde{\mathcal{D}}$. Then $\mathcal{D}$ and $\tilde{\mathcal{D}}$ both have the same volume. We write the transition probability from $\mathcal{D}$ to $\tilde{\mathcal{D}}$ as

$$\begin{aligned} T(\mathcal{D} \to \tilde{\mathcal{D}}) &= (\delta V) \min\left(1, \frac{\exp(-H(\tilde{\mathcal{D}}))}{\exp(-H(\mathcal{D}))}\right) \\ &= (\delta V) \min\left(1, \exp\left(-H(\tilde{\mathcal{D}}) + H(\mathcal{D})\right)\right) , \end{aligned} \tag{3.32}$$

where δV represents the volume of $\mathcal{D}$ and $\tilde{\mathcal{D}}$, and $\int_{\mathcal{D}} p(\boldsymbol{\theta}, \boldsymbol{r}) d\boldsymbol{\theta} d\boldsymbol{r} = \exp(-H(\mathcal{D}))$. If $(\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^*)$ is drawn from the joint density in (3.30), then $P((\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^*) \in \mathcal{D}) = \int_{\mathcal{D}} p(\boldsymbol{\theta}, \boldsymbol{r}) d\boldsymbol{\theta} d\boldsymbol{r} = \exp(-H(\mathcal{D}))$. Therefore,

$$\begin{aligned} &P((\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^*) \in \mathcal{D}, (\boldsymbol{\theta}^{(t)}, \boldsymbol{r}^{(t)}) \in \tilde{\mathcal{D}}) \\ &\quad = P((\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^*) \in \mathcal{D}) T(\mathcal{D} \to \tilde{\mathcal{D}}) \\ &\quad = \exp(-H(\mathcal{D}))\, (\delta V) \min\left(1, \exp\left(-H(\tilde{\mathcal{D}}) + H(\mathcal{D})\right)\right) \\ &\quad = (\delta V) \min\left(\exp(-H(\mathcal{D})), \exp\left(-H(\tilde{\mathcal{D}})\right)\right) \\ &\quad = P((\boldsymbol{\theta}^{(t-1)}, \boldsymbol{r}^*) \in \tilde{\mathcal{D}}, (\boldsymbol{\theta}^{(t)}, \boldsymbol{r}^{(t)}) \in \mathcal{D}) , \end{aligned} \tag{3.33}$$

where the last equality follows from the symmetry in the expression above it.

This procedure, while maintaining the stationary distribution through time-reversibility, introduces a host of complexities. Perhaps most importantly, tuning the many parameters needed in this process is inherently difficult. This motivated the development of an automatic procedure known as the No-U-Turn-Sampler (NUTS) Hoffman and Gelman (2011). During the warm-up phase of NUTS, $\boldsymbol{M}$ is estimated, as is the step size of the leapfrog method required to efficiently explore the phase space generated by the dynamical system. This warm-up estimation is achieves significant efficiency over the simple HMC algorithm described above by either explicitly avoiding a U-turn to a previously explored region or terminating after a pre-defined number of exploration steps. In this way, the algorithm is guaranteed to only explore new areas of the space. This efficient exploration results in typically faster convergence and higher effective sample sizes per iteration as compared to classical MCMC.

CHAPTER 4

Bayesian Calibration of Spatiotemporal Computer Models

4.1 Introduction

Melding of information from noisy data and computer model experiments is rapidly evolving into standard practice in diverse scientific applications. Computer models are versatile and often provide an accurate description of physical or mechanistic systems useful for furthering scientific understanding. These models incorporate a functional relationship controlled by physical parameters which, often, represent mechanistic structure. The optimal values of these parameters are unknown and need to be estimated from actual observations, which is an exercise often termed as the inverse problem. The inverse problem becomes challenging when the underlying model is computationally complex in nature and expensive to compute.

The focus of this work is on *emulating* and *calibrating* a spatiotemporal system. Emulation refers to the construction of a cheap statistical model that mimics the behavior of the true computer simulation, necessary for modeling with computationally expensive systems. The statistical emulator is useful as a proxy to quantify computer model input-output uncertainty. Calibration uses physical observations to uncover the most plausible parameter settings of the underlying emulator to solve the inverse problem for the true computer model. In this article, we focus on a Bayesian hierarchical model to build the statistical emulator and calibrator. As our motivating examples arise from the analysis of ordinary and partial nonlinear differential equations, we mention related work in the dynamic modeling community that also make use of hierarchical modeling. A hierarchical description of a physical process consists of a layered approach, whereby simpler conditional dependence structures specify complex dependencies and relationships. This modeling process can be factored into three

levels:

$$\begin{aligned}[\text{process, parameters} \mid \text{data}] \propto\ & [\text{data} \mid \text{process}, \text{parameters}] \\ & \times [\text{process} \mid \text{parameters}] \\ & \times [\text{parameters}] \ .\end{aligned}$$

At the top level, the probability model specifies the distribution of the data conditional on the physical process. The next level consists of the process model and can represent physical or mechanistic knowledge. The bottom level models uncertainty about parameters. In the context of modeling with partial differential equations (PDEs), the models of applied interest almost never admit analytic solutions and, hence, numerical approaches that involve discretizing and looping over the entire spatial domain are required. This is a computationally arduous task. A common approach is to discretize the mathematical equations and form a mechanistic-statistical state-space model to connect to observed data (Wikle et al., 2019; Hefley et al., 2017; Hooten et al., 2013).

However, this process often obscures the underlying mechanistic parameters because direct inference on these parameters is no longer the central goal. Other Bayesian approaches to modeling with differential equations include (Salter and Williamson, 2019; Cockayne et al., 2017) and (Wang et al., 2021), but they often center on probabilistic modeling of discretization error induced through the numerical solutions. If coarser meshes are used in the numerical solvers, computational gains can often be made at the expense of solution accuracy. In contrast, our methodology utilizes both a statistical-mechanistic and state-space model, but additionally augments with precise numerical evaluations where discretization error becomes negligible. The novelty is to retain the explicit connection to the underlying computer model or mechanistic parameters while still utilizing a structural framework to characterize spatiotemporal dynamics.

In the general analysis of deterministic models Kennedy and O'Hagan (2001) and Poole and Raftery (2000) popularized Bayesian calibration of computer experiments, which persists as an active field of research within both the machine learning and statistics communities; see Santner et al. (2003) for a textbook treatment and Shahriari et al. (2016) for a review article on the related field of Bayesian optimization. Calibration extensions for high-dimensional data later arose through the work of Higdon et al. (2004, 2008). Even so, these approaches scale poorly for simulations with large output due to the $\mathcal{O}(N^3)$ complexity of Gaussian process regression. Work to enable computationally efficient Bayesian emulation of systems with large temporal output has been

developed by Liu and West (2009) that combine Bayesian dynamic models with Gaussian processes. Farah et al. (2014) applied this methodology to calibrate an expensive simulation of influenza spread. In the following sections, we extend to the spatiotemporal setting by introducing another Gaussian process into the state-space model. Other work on combining Gaussian processes and state-space models may be found in the machine learning literature (Turner et al., 2010; Eleftheriadis et al., 2017), where the process is used to model the transition function rather than stochastic innovations.

The structure of this article is as follows. In Section 4.2 we develop the necessary background on Gaussian process regression and Bayesian state-space modeling before introducing spatiotemporal emulation. In Section 4.3, we apply our proposed method to calibrate various computer models arising in Bayesian analysis of ordinary and partial nonlinear differential equations as well as a simulator with no closed algebraic form. This simulation generates diffusion throughout a network of connected nodes with application to modeling phonological networks. These computer models have applications in population dynamics, infectious disease spread, psychology, and linguistics.

4.2 Methods

This section provides background material that forms the basis of our methodology. Specifically, Section 4.2.1 reviews Gaussian process regression and prediction. Section 4.2.2 discusses efficient methods for posterior inference in conjugate state-space models through recursive computation. We then detail our proposed methodology for Bayesian calibration. Section 4.2.3 details the construction of our spatiotemporal emulator, while Section 4.2.4 uses the emulator to solve the calibration problem with model bias. The remaining Section 4.2.5 details two strategies for scaling our model using parallel computing or reduced-rank Gaussian processes. By way of notation, define the following.

- Denote the computer input space by $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i$ is a d-dimensional vector. Let the collection of observed spatial locations be $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_S\}$.
- At spatial location $\boldsymbol{s}$ at time t, collect the outputs for each simulation setting as the N-dimensional vector $\boldsymbol{y}_t(\boldsymbol{s}) = (y_t(\boldsymbol{s}, \boldsymbol{x}_1), \ldots y_t(\boldsymbol{s}, \boldsymbol{x}_N))^\top$.
- Write the latent process $\boldsymbol{\phi}_t$ and field observations gathered over space as a $pS \times 1$ vector $\boldsymbol{\phi}_t =$

$(\phi_t(\boldsymbol{s}_1),\ldots,\phi_t(\boldsymbol{s}_S))^\top$ and $S\times 1$ vector $\boldsymbol{z}_t=(z_t(\boldsymbol{s}_1),\ldots,z_t(\boldsymbol{s}_S))^\top$, respectively.

4.2.1 Gaussian Process Regression

Consider a deterministic computer model that takes as input a d-dimensional vector $\boldsymbol{x}=(x_1,\ldots,x_d)^\top$ and outputs a scalar $y(\boldsymbol{x})$. To model this unknown relationship, we place a Gaussian process prior on the functional form so that $y(\boldsymbol{x})\sim\mathcal{GP}\big(\mu,\sigma^2C(\cdot;\boldsymbol{\beta})\big)$, where μ, σ^2, $C(\cdot;\boldsymbol{\beta})$, and $\boldsymbol{\beta}$ are the mean, variance, correlation function, and hyperparameters, respectively. It is often convenient to assume a squared exponential correlation function, of which we assume for the rest of this article. This correlation function takes the following form,

$$C(\boldsymbol{x},\boldsymbol{x}';\boldsymbol{\beta})=\exp\big(-\sum_{i=1}^{d}\beta_i(\boldsymbol{x}_i-\boldsymbol{x}_i')^2\big), \tag{4.1}$$

where the range parameter β_i controls the decay of correlation along the specific input dimension.

From standard identities, we can form the conditional distribution for a new input $\boldsymbol{x}^*$, in effect "learning" how the computer model behaves for unknown input values. These conditional identities result in a Gaussian predictive distribution $p\big(y(\boldsymbol{x}^*)|\boldsymbol{y}(\mathcal{X})\big)\sim\mathcal{N}(\bar{\mu}(\boldsymbol{x}^*),\bar{\sigma}^2(\boldsymbol{x}^*))$ with mean $\bar{\mu}(\boldsymbol{x}^*)=\mu+\boldsymbol{\gamma}^\top\boldsymbol{V}^{-1}(\boldsymbol{y}(\mathcal{X})-\mu\mathbf{1}_n)$ and variance $\bar{\sigma}^2(\boldsymbol{x}^*)=\sigma^2-\boldsymbol{\gamma}^\top\boldsymbol{V}^{-1}\boldsymbol{\gamma}/\sigma^2$. We define $\boldsymbol{\gamma}=\big(C(\boldsymbol{x}_1,\boldsymbol{x}^*;\boldsymbol{\beta}),\ldots,C(\boldsymbol{x}_N,\boldsymbol{x}^*;\boldsymbol{\beta})\big)^\top$ and $\boldsymbol{V}$ to be build through application of $C(\cdot;\boldsymbol{\beta})$ to $\mathcal{X}$. These predictive equations imply the Gaussian process interpolates the computer model at the design points as the variance shrinks to zero.

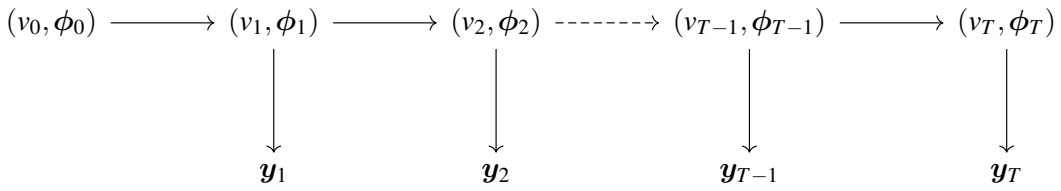
4.2.2 Bayesian State-Space Models

Bayesian state-space models (SSMs) – also known as Bayesian dynamic models – are a rich class of non-stationary time series models with a large literature on efficient computation (West and Harrison, 1997; Petris et al., 2009). There are three areas of interest when considering dynamic models: filtering, smoothing, and forecasting. Filtering refers to inference on the current latent state conditional on all current and past data. The most famous example of filtering is the Kalman filter (Kalman, 1960). Smoothing refers to retrodictive inference on the latent process conditional on the entire observed data. Finally, forecasting refers to inference on the hidden state at any time point beyond the current observation. These ideas are important when building an efficient sampling routine. The general structure of the state-space model in our methodology takes the

following form,

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{F}_t \boldsymbol{\phi}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}_N(0, v_t \boldsymbol{V}(\boldsymbol{\beta})) \\ \boldsymbol{\phi}_t &= \boldsymbol{G}_t \boldsymbol{\phi}_{t-1} + \boldsymbol{\tau}_t, \quad \boldsymbol{\tau}_t \sim \mathcal{N}_p(0, v_t \boldsymbol{W}(\boldsymbol{\psi})), \end{aligned} \tag{4.2}$$

where $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are matrices of order $N \times p$ and $p \times p$, respectively. The $N \times N$ and $p \times p$ correlation matrices $\boldsymbol{V}$ and $\boldsymbol{W}(\boldsymbol{\psi})$ are parameterized by $\boldsymbol{\beta}$ and $\boldsymbol{\psi}$, respectively. In contrast to standard Gaussian state-space models, we allow both the latent state $\boldsymbol{\phi}_t$ and v_t to obey Markovian dynamics. The directed acyclic graph (DAG) for this state-space model is



A discount learning model is useful for inducing time-varying dynamics of v_t and represents the relative learning rate or stability of a stochastic process over time (West and Harrison, 1997). The learning rate is commonly tuned using predictive criteria or maximization of the log-likelihood. Given discount factor $\delta_1 \in (0, 1]$, we assume a multiplicative model for the precision v_t^{-1} through

$$v_t^{-1} = \frac{\gamma_t}{\delta_1} v_{t-1}^{-1}, \tag{4.3}$$

where $v_{t-1}^{-1} \sim \mathcal{G}(n_{t-1}, d_{t-1})$ and $\gamma_t \sim \mathcal{B}(\delta_1 n_{t-1}, (1-\delta_1) n_{t-1})$. Standard distribution theory then shows $v_t^{-1} \sim \mathcal{G}(n_t^*, d_t^*)$, where $n_t^* = \delta_1 n_{t-1}$ and $d_t^* = \delta_1 d_{t-1}$. In this way, v_t^{-1} can be seen as evolving from v_{t-1}^{-1} times a random shock with expected value $\mathbb{E}[\gamma_t / \delta_1] = 1$. To complete the prior specification of the model given starting values $\boldsymbol{m}_0$, $\boldsymbol{M}_0$, n_0, and d_0, we write the conjugate joint prior as,

$$\begin{aligned} (\boldsymbol{\phi}_0, v_0^{-1}, \boldsymbol{\phi}_1, v_1^{-1}, \ldots, \boldsymbol{\phi}_T, v_T^{-1}) \sim{}& \mathcal{NG}(\boldsymbol{\phi}_0, v_0^{-1} \mid \boldsymbol{m}_0, \boldsymbol{M}_0, n_0, d_0) \\ & \times \prod_{t=1}^{T} \mathcal{NG}\left(\boldsymbol{\phi}_t, v_t^{-1} \mid \boldsymbol{\phi}_{t-1}, \boldsymbol{W}(\boldsymbol{\psi}), n_t^*, d_t^*\right), \end{aligned} \tag{4.4}$$

so the ensuing filtering operations of Algorithm 4 remain of Normal-gamma family. In Section 4.2.5, we also introduce a second discount factor to specify time-varying but deterministic dynamics for $\boldsymbol{W}(\boldsymbol{\psi})$.

For applied modeling, the computational efficiency arises through the latent Markov process on the regression parameters ϕ_t. This Markov structure enables efficient recursive computation of the full conditionals required for Markov Chain Monte Carlo (MCMC) sampling. This recursive computing strategy is known as the forward-filter-backward-sampling (FFBS) algorithm (Carter and Kohn, 1994; Frühwirth-Schnatter, 2006). In essence, this blocked Gibbs sampler relies on the Kalman filter for the forward pass of the algorithm, obtaining the distributional moments required to start the backward sampling. The backward sampling step is a stochastic version of the Kalman or Rauch–Tung–Striebel smoother (Särkkä, 2013). The FFBS algorithm is a blocked Gibbs sampler because draws made from the full latent trajectory $v_{0:T}, \phi_{0:T} | \boldsymbol{y}_{1:T}$ in entirety before proceeding with further parameter updates. This results in much faster convergence and lower autocorrelations in the posterior samples (Carter and Kohn, 1994; Scott, 2002). The FFBS algorithm for inferring latent parameters conditional upon $\boldsymbol{\beta}$ and $\boldsymbol{\psi}$ in model (4.2) is described below in Algorithm 1.

Algorithm 1 Forward-filter-backward-sample

Input: Data $\boldsymbol{y}_{1:T}$ and Kalman filter starting values n_0, d_0, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$
  Correlation matrices $\boldsymbol{V}$, $\boldsymbol{W}$
Output: Sample from posterior $p(v_{0:T}, \phi_{0:T} | \boldsymbol{y}_{1:T})$
function FFBS($\boldsymbol{y}_{1:T}, n_0, d_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{V}, \boldsymbol{W}(\psi)$)
  $\{n_t^*, d_t^*, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^T \leftarrow$ KalmanFilter($\boldsymbol{y}_{1:T}, n_0, d_0,$
    $\boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{V}, \boldsymbol{W}(\psi)$) ▷ Algorithm 4
  $v_{0:T}, \phi_{0:T} | \boldsymbol{y}_{1:T} \leftarrow$ BackwardSample($\{n_t^*, d_t^*, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^T$) ▷ Algorithm 5
  return $v_{0:T}, \phi_{0:T} | \boldsymbol{y}_{1:T}$
end function

4.2.3 Dynamic Spatiotemporal Emulation

We refer to our model as SSGP to emphasize the combination of Bayesian state-space and Gaussian process models. Provided a correlation function $C(\cdot;\boldsymbol{\beta})$ over the computer input space and cross-correlation function $\boldsymbol{K}(\cdot;\boldsymbol{\psi})$ over the spatial domain, our SSGP spatiotemporal computer emulator consists of a multivariate state-space model with Gaussian processes driving the stochastic innovations. The $N \times 1$ vector $\boldsymbol{f}_t$ contains the transition terms for the observation equation. For example, a simple specification is an autoregressive order p

transition structure at computer input $\boldsymbol{x}$ such that $\boldsymbol{f}_t(\boldsymbol{s},\boldsymbol{x}) = (y_{t-1}(\boldsymbol{s}),\ldots,y_{t-p}(\boldsymbol{s}))^\top$. Taking the correlation function in (4.1) as $C(\cdot;\boldsymbol{\beta})$, the spatiotemporal emulator model is

$$\begin{aligned} y_t(\boldsymbol{s},\boldsymbol{x}) &= \boldsymbol{f}_t(\boldsymbol{s},\boldsymbol{x})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) + \varepsilon_t(\boldsymbol{x}), \quad \varepsilon_t(\boldsymbol{x}) \sim \mathcal{GP}(0, v_t C(\cdot;\boldsymbol{\beta})) \\ \boldsymbol{\phi}_t(\boldsymbol{s}) &= \boldsymbol{G}_t(\boldsymbol{s})\boldsymbol{\phi}_{t-1}(\boldsymbol{s}) + \boldsymbol{\tau}_t(\boldsymbol{s}), \quad \boldsymbol{\tau}_t(\boldsymbol{s}) \sim \mathcal{GP}(0, v_t \boldsymbol{K}(\cdot;\boldsymbol{\psi})). \end{aligned} \tag{4.5}$$

Since $\boldsymbol{\tau}(\boldsymbol{s})$ is a p-dimensional vector-valued process, a convenient parameterization of a valid cross-correlation function is to consider again the univariate process $C(\cdot;\boldsymbol{\psi})$ and let $\boldsymbol{T}$ be a $p \times p$ positive definite matrix such that

$$\boldsymbol{K}\left(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\psi}\right) = C\left(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\theta}\right) \cdot \boldsymbol{T}, \tag{4.6}$$

where $\boldsymbol{\psi} = \{\boldsymbol{\theta}, h(\boldsymbol{T})\}$. Here the $p \times p$ matrix $\boldsymbol{T}$ is the covariance associated with $\boldsymbol{\tau}(\boldsymbol{s})$, while $C(\cdot;\boldsymbol{\psi})$ models the spatial association. The covariance matrix for the realized spatial process $\boldsymbol{\tau}_t$ is $\boldsymbol{W}(\boldsymbol{\psi}) = \boldsymbol{H} \otimes \boldsymbol{T}$, where $(\boldsymbol{H})_{ij} = C\left(\boldsymbol{s}_i,\boldsymbol{s}_j;\boldsymbol{\psi}\right)$ and $\otimes$ denotes the Kronecker product. The matrix $\boldsymbol{W}(\boldsymbol{\psi})$ is convenient to work with since $|\boldsymbol{W}(\boldsymbol{\psi})| = |\boldsymbol{H}|^S |\boldsymbol{T}|^p$ and $\boldsymbol{W}(\boldsymbol{\psi})^{-1} = \boldsymbol{H}^{-1} \otimes \boldsymbol{T}^{-1}$. A simple approach is to consider the autoregressive parameters comprising $\boldsymbol{\phi}(\boldsymbol{s})$ as independent, in which case $\boldsymbol{T} = \boldsymbol{I}$. A richer model specification involves an inverse Wishart prior for $\boldsymbol{T}$, so the full conditional distribution $p(\boldsymbol{T} \mid \cdot)$ remains inverse Wishart. Unfortunately, $\boldsymbol{T}$ (and hence $\boldsymbol{W}(\boldsymbol{\psi})$) is not identifiable due to the scaling factor v_t in (4.5). However, to each covariance matrix $\boldsymbol{T}$, there corresponds a unique correlation matrix obtained by the function

$$h(\boldsymbol{T}) = \left(\sigma_{ij} / \sqrt{\sigma_i^2 \sigma_j^2}\right)_{ij},$$

where σ_{ij} is the (i,j)-th element of $\boldsymbol{T}$. Since the value of $\boldsymbol{W}(\boldsymbol{\psi}) = \boldsymbol{H} \otimes h(\boldsymbol{T})$ is identifiable, one estimation approach is to reparameterize the model in terms of this non-identifiable covariance matrix $\boldsymbol{T}$, while focusing posterior inference on the identifiable correlation matrix $\boldsymbol{W}(\boldsymbol{\psi})$ after making the transformation $h(\boldsymbol{T})$. The addition of a non-identifiable parameter within Bayesian models to simplify calculations is well-established as parameter expansion (Liu and Wu, 1999; Gelman, 2004). To connect this to the state-space notation of Section 4.2.2, for a realization of computer design points at $\mathcal{X} = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$, we may form the $N \times N$ correlation

matrix $\boldsymbol{V}$ and write the autoregressive matrix of dimension $N \times p$ at spatial location $\boldsymbol{s}$ as

$$\boldsymbol{F}_t(\boldsymbol{s}) = \begin{pmatrix} y_{t-1}(\boldsymbol{s}, \boldsymbol{x}_1) & \cdots & y_{t-p}(\boldsymbol{s}, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ y_{t-1}(\boldsymbol{s}, \boldsymbol{x}_N) & \cdots & y_{t-p}(\boldsymbol{s}, \boldsymbol{x}_N). \end{pmatrix}$$

Given the observed spatial locations $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_S\}$, the $pS \times pS$ correlation matrix is $\boldsymbol{W}(\boldsymbol{\psi})$, so the realized computer emulation model is

$$\begin{aligned} \boldsymbol{y}_t(\boldsymbol{s}) &= \boldsymbol{F}_t(\boldsymbol{s})\boldsymbol{\phi}_t(\boldsymbol{s}) + \boldsymbol{\varepsilon}_t(\boldsymbol{s}), \quad \boldsymbol{\varepsilon}_t(\boldsymbol{s}) \sim \mathcal{N}(0, v_t \boldsymbol{V}(\boldsymbol{\beta})) \\ \boldsymbol{\phi}_t &= \boldsymbol{G}_t \boldsymbol{\phi}_{t-1} + \boldsymbol{\tau}_t, \quad \boldsymbol{\tau}_t \sim \mathcal{N}(0, v_t \boldsymbol{W}(\boldsymbol{\psi})), \end{aligned} \tag{4.7}$$

After introducing priors $p(\boldsymbol{\beta}, \boldsymbol{\psi})$ and $\mathcal{NG}\left(\boldsymbol{\phi}_0, v_0^{-1} | \boldsymbol{m}_0, \boldsymbol{M}_0, n_0, d_0\right)$, the posterior is

$$\begin{aligned} p(v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi} | \boldsymbol{y}_{1:T}) \propto\ & p(\boldsymbol{\beta}, \boldsymbol{\psi}) \mathcal{NG}\left(\boldsymbol{\phi}_0, v_0^{-1} \mid \boldsymbol{m}_0, \boldsymbol{M}_0, n_0, d_0\right) \\ & \times \prod_{t=1}^{T} \mathcal{NG}\left(\boldsymbol{\phi}_t, v_t^{-1} \mid \boldsymbol{\phi}_{t-1}, \boldsymbol{W}(\boldsymbol{\psi}), n_t^*, d_t^*\right) \\ & \times \prod_{t=1}^{T} \prod_{i=1}^{S} \mathcal{N}\left(\boldsymbol{y}_t(\boldsymbol{s}_i) \mid \boldsymbol{F}_t(\boldsymbol{s}_i)\boldsymbol{\phi}_t(\boldsymbol{s}_i), v_t \boldsymbol{V}(\boldsymbol{\beta})\right), \end{aligned} \tag{4.8}$$

where the hyperparameters of the priors arise from the Kalman filter Algorithm 4.

Draws from the posterior (4.8) enable forecasting computer model output for unknown input $\boldsymbol{\eta}$ using the emulator's predictive distribution. To build the predictive distribution, write the unknown output at spatial location $\boldsymbol{s}$ at input setting $\boldsymbol{\eta}$ as $y_t^*(\boldsymbol{s}, \boldsymbol{\eta})$. The joint distribution takes the form

$$\begin{bmatrix} y_t^*(\boldsymbol{s}, \boldsymbol{\eta}) \\ \boldsymbol{y}_t(\boldsymbol{s}) \end{bmatrix} \Big| \boldsymbol{f}_t(\boldsymbol{s}, \boldsymbol{\eta}), \boldsymbol{F}_t(\boldsymbol{s}), \boldsymbol{\eta}, v_t, \boldsymbol{\phi}_t, \boldsymbol{\beta} \Bigg] \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f}_t(\boldsymbol{s}, \boldsymbol{\eta})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) \\ \boldsymbol{F}_t(\boldsymbol{s})\boldsymbol{\phi}_t(\boldsymbol{s}) \end{bmatrix}, \begin{bmatrix} v_t & \boldsymbol{r}(\boldsymbol{\eta})^\top \\ \boldsymbol{r}(\boldsymbol{\eta}) & v_t \boldsymbol{V}(\boldsymbol{\beta}) \end{bmatrix} \right), \tag{4.9}$$

with $\boldsymbol{r}(\boldsymbol{\eta}) = (C(\boldsymbol{x}_1, \boldsymbol{\eta}; \boldsymbol{\beta}), \ldots, C(\boldsymbol{x}_N, \boldsymbol{\eta}; \boldsymbol{\beta}))^\top$. Like in Section 4.2.1, the emulator predictive density at input $\boldsymbol{\eta}$ is given after conditioning (4.9) so that $y_t^*(\boldsymbol{s}, \boldsymbol{\eta}) | \boldsymbol{\eta}, v_t, \boldsymbol{\phi}_t, \boldsymbol{\beta} \sim \mathcal{N}(\widetilde{\mu}_t(\boldsymbol{s}, \boldsymbol{\eta}), \widetilde{\sigma}_t^2(\boldsymbol{s}, \boldsymbol{\eta}))$, where

$$\begin{aligned} \widetilde{\mu}_t(\boldsymbol{s}, \boldsymbol{\eta}) &= \boldsymbol{f}_t(\boldsymbol{s}, \boldsymbol{\eta})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) + \boldsymbol{r}(\boldsymbol{\eta})^\top \boldsymbol{V}(\boldsymbol{\beta})^{-1} \left(\boldsymbol{y}_t(\boldsymbol{s}) - \boldsymbol{F}_t(\boldsymbol{s})\boldsymbol{\phi}_t(\boldsymbol{s})\right) \\ \widetilde{\sigma}_t^2(\boldsymbol{s}, \boldsymbol{\eta}) &= v_t(1 - \boldsymbol{r}(\boldsymbol{\eta})^\top \boldsymbol{V}(\boldsymbol{\beta})^{-1} \boldsymbol{r}(\boldsymbol{\eta})). \end{aligned} \tag{4.10}$$

Additionally, we can make predictions at arbitrary spatial location $\boldsymbol{s}^*$ through the spatial process $\boldsymbol{\tau}_t$. Through similar analysis, $\boldsymbol{\phi}_t(\boldsymbol{s}^*)|\boldsymbol{\phi}_{t-1}(\boldsymbol{s}^*), \boldsymbol{\phi}_t, \boldsymbol{\phi}_{t-1}, v_t, \boldsymbol{\psi} \sim \mathcal{N}_p(\boldsymbol{\lambda}_t(\boldsymbol{s}^*), \boldsymbol{\Lambda}_t(\boldsymbol{s}^*))$ where

$$\begin{aligned} \boldsymbol{\lambda}_t(\boldsymbol{s}^*) &= \boldsymbol{\phi}_{t-1}(\boldsymbol{s}^*) + \boldsymbol{k}(\boldsymbol{s}^*)^\top \boldsymbol{W}(\boldsymbol{\psi})^{-1}(\boldsymbol{\phi}_t - \boldsymbol{\phi}_{t-1}) \\ \boldsymbol{\Lambda}_t(\boldsymbol{s}^*) &= v_t \boldsymbol{K}(\boldsymbol{s}^*, \boldsymbol{s}^*; \boldsymbol{\psi}) - v_t \boldsymbol{k}(\boldsymbol{s}^*)^\top \boldsymbol{W}(\boldsymbol{\psi})^{-1} \boldsymbol{k}(\boldsymbol{s}^*), \end{aligned} \tag{4.11}$$

where the matrix $\boldsymbol{k}(\boldsymbol{s}^*)^\top = [\boldsymbol{K}(\boldsymbol{s}_1, \boldsymbol{s}^*; \boldsymbol{\psi}) : \cdots : \boldsymbol{K}(\boldsymbol{s}_S, \boldsymbol{s}^*; \boldsymbol{\psi})]$ is $p \times pS$. This general idea has computational implications for dimensionality reduction, shown in Section 4.2.5.

4.2.4 Calibration with Computer Model Bias

The above emulator is now used to solve the spatiotemporal calibration problem. Often a computer model may not adequately describe observed field data over all spatial locations and time points, so we include a spatiotemporally-varying misspecification term that captures bias between the observed field data and emulator. Our calibration model with dynamic bias correction has the following form,

$$\begin{aligned} z_t(\boldsymbol{s}) &= y_t^*(\boldsymbol{s}, \boldsymbol{\eta}) + u_t(\boldsymbol{s}) + \varepsilon_t^z(\boldsymbol{s}), \quad \varepsilon_t^z(\boldsymbol{s}) \sim \mathcal{N}(0, v_t) \\ u_t(\boldsymbol{s}) &= u_{t-1}(\boldsymbol{s}) + \omega_t(\boldsymbol{s}), \quad \omega_t(\boldsymbol{s}) \sim \mathcal{GP}(0, v_t C(\cdot; \boldsymbol{\rho})), \end{aligned} \tag{4.12}$$

where $\boldsymbol{U}(\boldsymbol{\rho})$ is the $S \times S$ correlation matrix built through $C(\cdot; \boldsymbol{\rho})$. Introducing priors

$$\mathcal{U}(\boldsymbol{\eta}|0,1), \mathcal{NG}(\boldsymbol{\phi}_0, v_0^{-1}|\boldsymbol{m}_0, \boldsymbol{M}_0, n_0, d_0),$$

and conditioning on the emulator of the previous section, the posterior is

$$\begin{aligned} p(\boldsymbol{\eta}, \boldsymbol{\rho}, v_{0:T}, \boldsymbol{u}_{0:T}|v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi}, \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}) \propto\ & p(\boldsymbol{\rho})\mathcal{U}(\boldsymbol{\eta}|0,1)\mathcal{NG}(\boldsymbol{u}_0, v_0^{-1}|\boldsymbol{m}_0, \boldsymbol{M}_0, n_0, d_0) \\ & \times \prod_{t=1}^{T} \mathcal{NG}(\boldsymbol{u}_t, v_t^{-1}|\boldsymbol{u}_{t-1}, \boldsymbol{U}(\boldsymbol{\rho}), n_t^*, d_t^*) \\ & \times \prod_{t=1}^{T}\prod_{i=1}^{S} \mathcal{N}\left(z_t(\boldsymbol{s}_i)|\widetilde{\mu}_t(\boldsymbol{\eta}, \boldsymbol{s}_i), \widetilde{\sigma}_t(\boldsymbol{\eta}, \boldsymbol{s}_i)\right), \end{aligned} \tag{4.13}$$

Algorithm 2 Sampler for emulation

Given: Data $\boldsymbol{y}_{1:T}$, discount factor δ_1, and tuning parameters $\varepsilon_1, \varepsilon_2$

Initialize: Gaussian process hyperparameters $\boldsymbol{\beta}^{(0)}$, $\boldsymbol{\psi}^{(0)}$

Kalman filter starting values n_0, d_0, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$

Output: posterior samples from $p(v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi} | \boldsymbol{y}_{1:T})$

for $i = 1$ to $No.\ samples$ **do**

$\boldsymbol{\beta}^{(i)}, \boldsymbol{\psi}^{(i)}, v_{0:T}^{(i)}, \boldsymbol{\phi}_{0:T}^{(i)} \leftarrow$ `updateEmulator`$(\boldsymbol{y}_{1:T}, \boldsymbol{\beta}^{(i-1)}, \boldsymbol{\psi}^{(i-1)}, \boldsymbol{\phi}_{0:T}^{(i-1)}, v_{0:T}^{(i-1)})$

end for

function UPDATEEMULATOR$(\boldsymbol{y}_{1:T}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\phi}_{0:T}, v_{0:T})$

update spatial process hyperparameters

Set $SS = \sum_{t=1}^{T} \sum_{i=1}^{S} (\boldsymbol{\phi}_t(\boldsymbol{s}_i) - \boldsymbol{G}_t(\boldsymbol{s}_i)\boldsymbol{\phi}_{t-1}(\boldsymbol{s}_i))(\boldsymbol{\phi}_t(\boldsymbol{s}_i) - \boldsymbol{G}_t(\boldsymbol{s}_i)\boldsymbol{\phi}_{t-1}(\boldsymbol{s}_i))^\top$

Draw $\boldsymbol{T}^{-1} \sim \mathcal{W}(v_0 + T/2, \boldsymbol{T}_0 + SS/2)$

$\boldsymbol{\theta}^* \leftarrow \exp(\log(\boldsymbol{\theta}) + \mathcal{N}(0, \varepsilon_1^2 \boldsymbol{I}))$

Set $\boldsymbol{\psi} = \{\boldsymbol{\theta}, h(\boldsymbol{T})\}$, $\boldsymbol{\psi}^* = \{\boldsymbol{\theta}^*, h(\boldsymbol{T})\}$ and Jacobian $J(\boldsymbol{\theta}) = \prod_{i=1}^{d} \boldsymbol{\theta}_i^{-1}$

Compute correlation matrix $\boldsymbol{W}(\boldsymbol{\psi})$ through $\boldsymbol{K}(\cdot; \boldsymbol{\psi})$ and $\boldsymbol{W}(\boldsymbol{\psi}^*)$ through $\boldsymbol{K}(\cdot; \boldsymbol{\psi}^*)$

With probability $\max\left\{ \frac{\prod_t \mathcal{N}(\boldsymbol{\phi}_t | \boldsymbol{\phi}_{t-1}, v_t \boldsymbol{W}(\boldsymbol{\psi}^*)) p(\boldsymbol{\theta}^*) \mathcal{J}(\boldsymbol{\theta})}{\prod_t \mathcal{N}(\boldsymbol{\phi}_t | \boldsymbol{\phi}_{t-1}, v_t \boldsymbol{W}(\boldsymbol{\psi})) p(\boldsymbol{\theta}) \mathcal{J}(\boldsymbol{\theta}^*)}, 1 \right\}$, $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi}^*$

update computer process hyperparameters

$\boldsymbol{\beta}^* \leftarrow \exp(\log(\boldsymbol{\beta}) + \mathcal{N}(0, \varepsilon_2^2 \boldsymbol{I}))$

Compute correlation matrix $\boldsymbol{V}(\boldsymbol{\beta})$ through $C(\cdot; \boldsymbol{\beta})$ and $\boldsymbol{V}^*(\boldsymbol{\beta}^*)$ through $C(\cdot; \boldsymbol{\beta}^*)$

Using the likelihood from (4.8) and Jacobian $J(\boldsymbol{\beta}) = \prod_{i=1}^{d} \boldsymbol{\beta}_i^{-1}$

With probability $\max\left\{ \frac{\mathcal{L}(\boldsymbol{\beta}^*) p(\boldsymbol{\beta}^*) \mathcal{J}(\boldsymbol{\beta})}{\mathcal{L}(\boldsymbol{\beta}) p(\boldsymbol{\beta}) \mathcal{J}(\boldsymbol{\beta}^*)}, 1 \right\}$, $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}^*$

$v_{0:T}, \boldsymbol{\phi}_{0:T} \leftarrow$ `FFBS`$(\boldsymbol{y}_{1:T}, n_0, d_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{V}(\boldsymbol{\beta}), \boldsymbol{W}(\boldsymbol{\psi}))$ ▷ Algorithm 1

return $\{\boldsymbol{\beta}, \boldsymbol{\psi}, v_{0:T}, \boldsymbol{\phi}_{0:T}\}$

end function

where $\widetilde{\mu}_t(\boldsymbol{\eta}, \boldsymbol{s})$ and $\widetilde{\sigma}_t(\boldsymbol{\eta}, \boldsymbol{s})$ are specified through

$$\begin{aligned}\widetilde{\mu}_t(\boldsymbol{s}, \boldsymbol{\eta}) &= \boldsymbol{f}_t(\boldsymbol{s}, \boldsymbol{\eta})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) + u_t(\boldsymbol{s}) + \boldsymbol{r}(\boldsymbol{\eta})^\top \boldsymbol{V}(\boldsymbol{\beta})^{-1} (\boldsymbol{y}_t(\boldsymbol{s}) - \boldsymbol{F}_t(\boldsymbol{s})\boldsymbol{\phi}_t(\boldsymbol{s})) \\ \widetilde{\sigma}_t^2(\boldsymbol{s}, \boldsymbol{\eta}) &= v_t + v_t \left(1 - \boldsymbol{r}(\boldsymbol{\eta})^\top \boldsymbol{V}(\boldsymbol{\beta})^{-1} \boldsymbol{r}(\boldsymbol{\eta})\right).\end{aligned} \tag{4.14}$$

The primary interest in the calibration phase is often the marginal distribution

$$\begin{aligned}p(\boldsymbol{\eta}, \boldsymbol{\rho}, \mathbf{v}_{0:T}, \boldsymbol{u}_{0:T} | \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}) &= \int p(\boldsymbol{\eta}, \boldsymbol{\rho}, \mathbf{v}_{0:T}, v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi} | \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}) dv_{0:T} d\boldsymbol{\phi}_{0:T} d\boldsymbol{\beta} d\boldsymbol{\psi} \\ &= \int p(\boldsymbol{\eta}, \boldsymbol{\rho}, \mathbf{v}_{0:T}, \boldsymbol{u}_{0:T} \big| v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi}, \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}) \\ &\qquad \times \underbrace{p(v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi} | \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T})}_{\text{Modularize}} dv_{0:T} d\boldsymbol{\phi}_{0:T} d\boldsymbol{\beta} d\boldsymbol{\psi}.\end{aligned} \tag{4.15}$$

A pragmatic approach to sampling from (4.15) relies on the posterior in (4.13) and the notion of *Bayesian Modularization* (Bayarri et al., 2009). Modularization replaces the underlined expression with the lower dimensional distribution $p(v_{0:T}, \boldsymbol{\phi}_{0:T}, \boldsymbol{\beta}, \boldsymbol{\psi} | \boldsymbol{y}_{1:T})$, which is readily sampled through FFBS and Metropolis-Hastings steps of Algorithm 2. Once these samples are available, we use Algorithm 3 to sample from (4.13), thus achieving draws from the desired marginal 4.15. This is justified both computationally and theoretically, as the observed field data should not influence our construction of a statistical emulator. Marginal distribution (4.15) is thus a principled way to account for emulator uncertainty when calibrating to observed data.

There exists a final opportunity to employ Bayesian modularization to prevent the model discrepancy term $\boldsymbol{u}_t$ from interfering with the calibration inference. In essence, this modularization treats the emulator as an unbiased model of the field data. After fully sampling calibration parameters, each update of the dynamic bias term conditions upon a draw from the parameter posterior to model the discrepancy between the field data and emulator predictive distribution. This is then a straightforward application of the function in Algorithm 3.

4.2.5 Strategies for Scalability

This section demonstrates strategies for scaling our framework to the setting of large spatiotemporal computer model learning. The first strategy is purely computational, where we eschew spatial structure for a fully heterogeneous conjugate Bayesian model amenable to embarrassingly parallel computation. An alternative strategy makes use of a rich collection of low rank models to preserve approximate spatial structure (Gelfand

Algorithm 3 Sampler for calibration with computer model bias

Given: Data $\boldsymbol{z}_{1:T}$ and tuning parameter ε_3

Initialize: Gaussian process hyperparameters $\boldsymbol{\rho}^{(0)}$

\# Modularization step for emulation

Execute Algorithm 2 and save MCMC samples

for $i = 1$ to *No. samples* **do**

    \# update calibration parameters

    Set $\boldsymbol{\eta}^{(i)} \leftarrow \boldsymbol{\eta}^{(i-1)}$

    $\boldsymbol{\zeta} \leftarrow \text{logit}^{-1}(\boldsymbol{\eta}^{(i)})$

    $\boldsymbol{\eta}^* \leftarrow \text{logit}(\boldsymbol{\zeta} + \mathcal{N}(0, \varepsilon_3^2 \boldsymbol{I}))$

    \# using the likelihood (4.13)

    With probability $\max\left\{\frac{\mathcal{L}(\boldsymbol{\eta}^*)p(\boldsymbol{\eta}^*)\mathcal{J}(\boldsymbol{\eta})}{\mathcal{L}(\boldsymbol{\eta})p(\boldsymbol{\eta})\mathcal{J}(\boldsymbol{\eta}^*)}, 1\right\}, \boldsymbol{\eta}^{(i)} \leftarrow \boldsymbol{\eta}^*$

    \# conditional on $\boldsymbol{\eta}^{(i)}$, use SSM structure of (4.12) to apply FFBS

    Set $\boldsymbol{z}^*_{1:T} = (y^*_{1:T}(\boldsymbol{s}_1, \boldsymbol{\eta}), \ldots, y^*_{1:T}(\boldsymbol{s}_S, \boldsymbol{\eta}))^\top$ ▷ stack over space in (4.10)

    $\boldsymbol{\rho}^{(i)}, \boldsymbol{v}^{(i)}_{0:T}, \boldsymbol{u}^{(i)}_{0:T} \leftarrow \texttt{updateDynamicBias}(\boldsymbol{z}^*_{1:T}, \boldsymbol{\rho}^{(i-1)}, \boldsymbol{u}^{(i-1)}_{0:T}, \boldsymbol{v}^{(i-1)}_{0:T})$

end for

function UPDATEDYNAMICBIAS($\boldsymbol{z}^*_{1:T}, \boldsymbol{\rho}, \boldsymbol{u}_{0:T}, \boldsymbol{\nu}_{0:T}$)

    \# update bias process hyperparameters

    $\boldsymbol{\rho}^* \leftarrow \exp(\log(\boldsymbol{\rho}) + \mathcal{N}(0, \varepsilon_3^2 \boldsymbol{I}))$

    Compute correlation matrix $\boldsymbol{U}(\boldsymbol{\rho})$ through $C(\cdot; \boldsymbol{\rho})$ and $\boldsymbol{U}^*(\boldsymbol{\rho}^*)$ through $C(\cdot; \boldsymbol{\rho}^*)$

    \# using likelihood in (4.14)

    With probability $\max\left\{\frac{\prod_t \mathcal{N}(\boldsymbol{u}_t | \boldsymbol{u}_{t-1}, \boldsymbol{v}_t \boldsymbol{U}^*(\boldsymbol{\rho}^*))p(\boldsymbol{\rho}^*)\mathcal{J}(\boldsymbol{\rho})}{\prod_t \mathcal{N}(\boldsymbol{u}_t | \boldsymbol{u}_{t-1}, \boldsymbol{v}_t \boldsymbol{U}(\boldsymbol{\rho}))p(\boldsymbol{\rho})\mathcal{J}(\boldsymbol{\rho}^*)}, 1\right\}, \boldsymbol{\rho} \leftarrow \boldsymbol{\rho}^*$

    $\boldsymbol{v}_{0:T}, \boldsymbol{u}_{0:T} \leftarrow \texttt{FFBS}(\boldsymbol{z}^*_{1:T}, n_0, d_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{I}, \boldsymbol{U}(\boldsymbol{\rho}))$ ▷ Algorithm 1

    return $\{\boldsymbol{\rho}, \boldsymbol{v}_{0:T}, \boldsymbol{u}_{0:T}\}$

end function

et al., 2010; Quiñonero-Candela and Rasmussen, 2005).

4.2.5.1 Seemingly Unrelated Spatiotemporal Models

We first present a scalable emulator inspired from the time series literature on seemingly unrelated time series equations (Zellner and Ando, 2010; Wooldridge, 2002). The assumption of the model is that error terms are independent across observations, but may have cross-equation correlations within observations due to the computer input space. In other words, each spatial location is modeled independently, while each site shares the influence from the computer input space. Using the notation of Section 4.2.3, the fully heterogeneous emulator with no underlying latent spatial structure is

$$\begin{aligned} y_t(\boldsymbol{s},\boldsymbol{x}) &= \boldsymbol{f}_t(\boldsymbol{s},\boldsymbol{x})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) + \boldsymbol{\varepsilon}_t(\boldsymbol{x}), \quad \boldsymbol{\varepsilon}_t(\boldsymbol{x}) \sim \mathcal{GP}\left(0, v_t(\boldsymbol{s})C(\cdot;\boldsymbol{\beta})\right) \\ \boldsymbol{\phi}(\boldsymbol{s}) &= \boldsymbol{G}_t(\boldsymbol{s})\boldsymbol{\phi}_{t-1}(\boldsymbol{s}) + \boldsymbol{\tau}_t(\boldsymbol{s}), \quad \boldsymbol{\tau}_t(\boldsymbol{s}) \sim \mathcal{N}\left(0, v_t(\boldsymbol{s})\boldsymbol{W}(\boldsymbol{s})\right), \end{aligned} \tag{4.16}$$

where $\boldsymbol{W}(\boldsymbol{s}) = \text{Corr}(\boldsymbol{\phi}_t(\boldsymbol{s}))$. Such model structure is amenable to embarrassingly parallel computation, where S applications of FFBS run during the MCMC sampling in Algorithm 2. We provide this model implementation in our code as well as the fully spatial version to provide comparison. As this heterogeneous spatial model is a "forest" of one-dimensional state-space Gaussian process emulators, we make use of the univariate discount factor specification to specify time-varying but deterministic dynamics for $\boldsymbol{W}(\boldsymbol{s})$ (Liu and West, 2009).

4.2.5.2 Low Rank Gaussian Processes

To retain the latent spatial structure but facilitate scalable computation, we replace the process $\boldsymbol{\phi}_t$ with a low-rank approximation such as the Gaussian predictive process (Banerjee et al., 2008; Banerjee, 2017). More precisely, we consider a smaller set of sites in the domain of interest called "knots," say $\mathcal{S}^* = \{\boldsymbol{s}_1^*, \ldots, \boldsymbol{s}_{S^*}^*\}$ where $S^* \ll S$. Now let $\boldsymbol{\tau}_t^*$ be a realization of $\boldsymbol{\tau}_t(\boldsymbol{s})$ over $\mathcal{S}^*$. That is, $\boldsymbol{\tau}_t^*$ is $pS^* \times 1$ with entries $\boldsymbol{\tau}_t(\boldsymbol{s}_i^*)$ and $\boldsymbol{\tau}_t^* \sim \mathcal{N}(0, v_t\boldsymbol{K}^*(\boldsymbol{\psi}))$, where $\boldsymbol{K}^*(\boldsymbol{\psi})$ is the associated $pS^* \times pS^*$ correlation matrix with entries $\boldsymbol{K}(\boldsymbol{s}_i^*, \boldsymbol{s}_j^*; \boldsymbol{\psi})$. The spatial interpolant that leads to the Kriging estimate at spatial location $\boldsymbol{s}_0$ is given by

$$\boldsymbol{\tau}(\boldsymbol{s}_0) = \mathbb{E}[\boldsymbol{\tau}_t(\boldsymbol{s}_0)|\boldsymbol{\tau}_t^*] = \boldsymbol{k}(\boldsymbol{s}_0;\boldsymbol{\psi})^\top \boldsymbol{K}^*(\boldsymbol{\psi})^{-1}\boldsymbol{\tau}_t^*. \tag{4.17}$$

This interpolator defines a spatial process $\widetilde{\boldsymbol{\tau}}_t(\boldsymbol{s}) \sim \mathcal{GP}(0, \boldsymbol{K}^*(\cdot;\boldsymbol{\psi}))$. The cross-correlation function of this low-rank process takes the form

$$\boldsymbol{K}^*(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\psi}) = \boldsymbol{k}(\boldsymbol{s};\boldsymbol{\psi})^\top \boldsymbol{K}^*(\boldsymbol{\psi})^{-1}\boldsymbol{k}(\boldsymbol{s}';\boldsymbol{\psi}), \tag{4.18}$$

with $\boldsymbol{k}(\boldsymbol{s};\boldsymbol{\psi})^\top$ a $p \times pS^*$ matrix whose j-th column is $\boldsymbol{K}(\boldsymbol{s},\boldsymbol{s}_j^*;\boldsymbol{\psi})$ for $j \in \{1,\ldots,S^*\}$. Replacing $\boldsymbol{\tau}_t(\boldsymbol{s})$ in (4.5) with $\widetilde{\boldsymbol{\tau}}_t(\boldsymbol{s}) + \widetilde{\boldsymbol{\varepsilon}}_t(\boldsymbol{s})$ completes the scalable emulator specification. The term $\widetilde{\boldsymbol{\varepsilon}}_t(\boldsymbol{s})$ is introduced to account for a new source of error with distributional form, $\mathcal{N}_p(0, \boldsymbol{\Delta}(\boldsymbol{s}))$ where $\boldsymbol{\Delta}(\boldsymbol{s}) = \boldsymbol{K}(\boldsymbol{s},\boldsymbol{s};\boldsymbol{\psi}) - \boldsymbol{k}(\boldsymbol{s};\boldsymbol{\psi})^\top \boldsymbol{K}^*(\boldsymbol{\psi})^{-1}\boldsymbol{k}(\boldsymbol{s};\boldsymbol{\psi})$ is diagonal $pS^* \times pS^*$; see (Finley et al., 2012) for discussion of this *bias-corrected* predictive processes. The additional source of error arises as the conditional expectation projects to a new Hilbert space and induces additional over-smoothing. This projection reduces the computational complexity during sampling from $\mathcal{O}(S^3)$ to $\mathcal{O}(S^{*3})$. To summarize, the scalable Gaussian predictive process dynamic emulator on knot locations $\mathcal{S}^*$ is

$$\begin{aligned} y_t(\boldsymbol{s},\boldsymbol{x}) &= \boldsymbol{f}_t(\boldsymbol{s},\boldsymbol{x})^\top \boldsymbol{\phi}_t(\boldsymbol{s}) + \varepsilon_t(\boldsymbol{x}),\ \varepsilon_t(\boldsymbol{x}) \sim \mathcal{GP}(0, v_t C(\cdot;\boldsymbol{\beta})) \\ \boldsymbol{\phi}_t(\boldsymbol{s}) &= \boldsymbol{G}_t(\boldsymbol{s})\boldsymbol{\phi}_{t-1}(\boldsymbol{s}) + \widetilde{\boldsymbol{\tau}}_t(\boldsymbol{s}),\ \widetilde{\boldsymbol{\tau}}_t(\boldsymbol{s}) \sim \mathcal{GP}(0, v_t \boldsymbol{K}^{**}(\cdot;\boldsymbol{\psi})), \end{aligned} \tag{4.19}$$

where $\boldsymbol{K}^{**}(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\psi}) = \boldsymbol{K}^*(\boldsymbol{s},\boldsymbol{s}';\boldsymbol{\psi}) + \mathbf{1}(\boldsymbol{s}=\boldsymbol{s}')\boldsymbol{\Delta}(\boldsymbol{s})$. Sampling from the posterior of (4.19) is a straightforward modification of Algorithm 2 using the new correlation function $\boldsymbol{K}^{**}$. Lastly, we mention knot selection can be an important consideration in applied modeling; see (Finley et al., 2009) or (Banerjee et al., 2014, Ch. 12) for various selection criteria. However, in our applied analyses, we found the calibration robust to knot placement.

4.2.6 Model Scoring

For model comparison and scoring purposes, we make use of a criteria based upon the predictive distribution of independently replicated data developed by Gneiting and Raftery (2007). To develop notation for the scoring rule, let $z_t^{\text{rep}}(\boldsymbol{s}_j)$ denote the replicate from the posterior predictive distribution for $z_t(\boldsymbol{s}_j)$. For all space-time coordinates, collect the $S \times T$ replicated data vector as $\boldsymbol{z}^{\text{rep}}$. Joining all emulation and calibration model

parameters into $\boldsymbol{\Theta}$, the posterior predictive distribution for $\boldsymbol{z}^{\text{rep}}$ is

$$p\left(\boldsymbol{z}^{\text{rep}}|\boldsymbol{z}_{1:T}\right) = \int p\left(\boldsymbol{z}^{\text{rep}}|\boldsymbol{z}_{1:T},\boldsymbol{\Theta}\right)p(\boldsymbol{\Theta}|\boldsymbol{z}_{1:T},\boldsymbol{y}_{1:T})d\boldsymbol{\Theta} \tag{4.20}$$

The GRS score is defined as

$$\text{GRS} = -\sum_{t=1}^{T}\sum_{j=1}^{S}\left(\frac{z_t(\boldsymbol{s}_j)-\mu_{tj}^{\text{rep}}}{\sigma_{tj}^{\text{rep}}}\right)^2 - \sum_{t=1}^{T}\sum_{j=1}^{S} 2\log\left(\sigma_{tj}^{\text{rep}}\right), \tag{4.21}$$

where μ_{tj}^{rep} and σ_{tj}^{rep} the mean and standard deviation of $z_t^{\text{rep}}(\boldsymbol{s}_j)$, respectively. Notice GRS is easily evaluated after obtaining posterior samples. The GRS depends only on the first and second moments of the predictive distribution for the replicated data and penalizes departure of replicated means from the corresponding observed values. In this way, parameter uncertainty is incorporated in the scoring. Models with higher GRS are desired.

4.3 Applications

In this section, we apply our methodology to a disparate group of spatiotemporal computer models arising in the analysis of ordinary and partial nonlinear differential equations inference and algorithmic network diffusion processes. Differential equations are specified through mathematical functions and are thus amenable to a discretization and mechanistic state-space model construction. However, the algorithmic network diffusion does not admit a similar discretization, as the functional relationship is purely algorithmic. We demonstrate that even without a highly parameterized structural state-space model, our method is still capable of parameter inference and learning of the computer model. These computer models are build with the `ReacTran` package (Soetaert and Meysman, 2012) and `spreadr` (Siew, 2019) package to demonstrate the ability of our methodology to capture and calibrate real external computer model behavior. We specifically chose scientific models from the literature where the calibration problem is ignored, thereby demonstrating a black-box utility of our framework where the applied researcher may focus on the forward modeling process. The first example is a pedagogical one where an ecological model is used to mathematically capturing changing population dynamics. The second application is to a set of nonlinear partial differential equations describing infectious disease dynamics. The third is calibration of a model developed within the linguistics community to describe a network diffusion

mechanism could be extended to account for higher false alarm rates for low clustering coefficient words in a false memory task. The final set of simulations demonstrated how spreading activation could be applied to a semantic network to account for semantic priming effects. Once these model replicates are available, our method solves the inverse problem while providing uncertainty quantification.

In all of our applications, we utilize a space-filling Latin hyper cube design (Santner et al., 2003) across the parameter space for a few training runs. Additionally, for all examples we make use of an AR(1) state-space model with non-informative uniform $\mathcal{U}(0,1)$ priors on the normalized calibration parameters. We fix the discount factor to $\delta_1 = 0.95$, though results were empirically robust to different values. By choosing an AR(1) process, we enforce a maximally parsimonious structure and achieve good performance in all applications. In building more structure into the state-space equations through structural time series methods (Petris et al., 2009), there exist further opportunity to tailor a bespoke emulator using domain-specific knowledge. To validate our methods, we employ a holdout testing dataset that is not included in the original computer model training runs.

4.3.1 Pedagogical Example: Calibrating an Inexpensive Computer Model

Our first application is towards calibrating a multivariate dynamical system specified through a set of coupled ordinary differential equations. This may be considered a multivariate statistical modeling problem in lieu of spatial. The numerical solution to this dynamical system – which we call our computer model – is cheap to evaluate. This facilitates a direct comparison between our methodology and repeatedly solving the differential equations at different parameter settings within an MCMC procedure. Specifically, we make use of the probabilistic programming language Stan (Carpenter et al., 2017; Stan Development Team, 2021) to repeatedly solve this system using Runge-Kutta methods. A log-normal likelihood is centered on the log-numerical solutions and used to connect to observed field data. This is similar to the approach of (Frankenburg and Banerjee, 2022) for modeling multivariate epidemiological time series, but we demonstrate that the emulation/calibration methodology can achieve similar performance but with orders of magnitude less computer model evaluations.

To motivate the ecological computer model, we overview the Lotka-Volterra equations, which describe a mechanism by which predator and prey population sizes oscillate. The equations generating this oscillatory

system are

$$\begin{aligned} \frac{du_t}{dt} &= \eta_1 u_t - \eta_2 u_t v_t \\ \frac{dv_t}{dt} &= -\eta_3 v_t + \eta_4 u_t v_t. \end{aligned} \tag{4.22}$$

For our noisy field observations, we utilize the data recorded on the Canadian lynx and snowshoe hare population sizes (Hewitt, 1917). Within (4.22), η_1 represents the growth rate of the prey population, while η_2 controls the rate of prey population shrinkage relative to the product of the population sizes. Likewise, η_3 represents the rate of population loss within the predator population, and η_4 controls the growth relative to the product of both populations. To formulate weekly informative priors for these parameters, we take $\eta_1, \gamma \sim \mathcal{N}(1, .5)$ to center the rate of change of both populations around their current sizes. Additionally, we take $\eta_2, \delta \sim \mathcal{N}(0.05, 0.05)$ to restrain the growth and shrinkage of the respective populations to be a small percentage of the product of the two. We generate training data generated from these priors over a Latin hypercube design as the collection of computer experiments, shown in Figure 4.1. Shown as black dots are the observed field data.

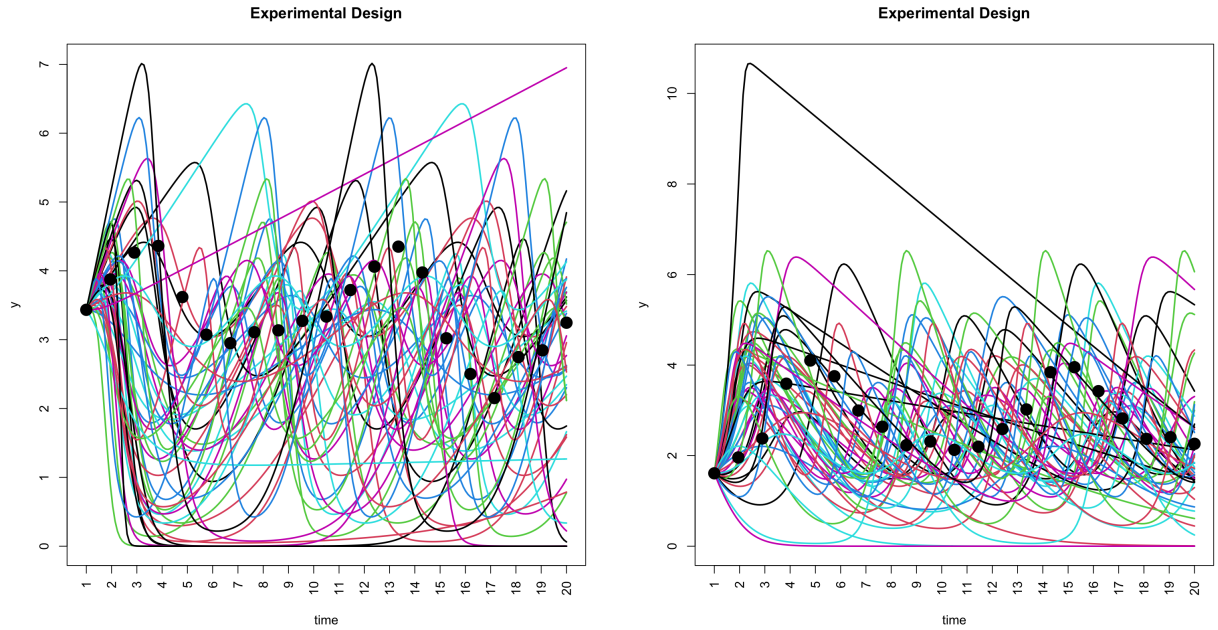


Figure 4.1: Computer model training runs and field data for Lotka-Volterra mechanism

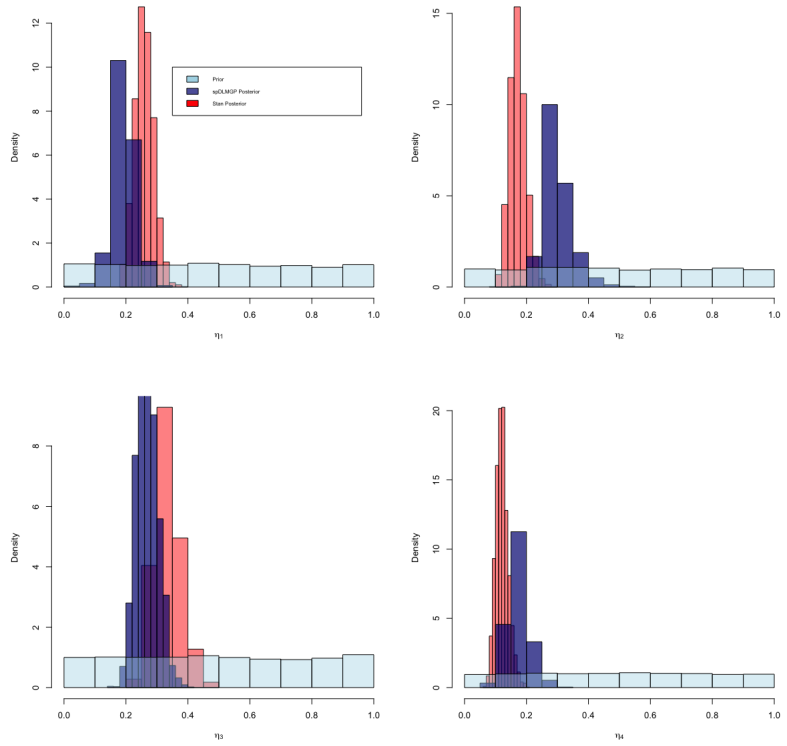


Figure 4.2: SSGP vs. Stan posteriors

To apply our methodology, let $\boldsymbol{y}_t = (u_t, v_t)^\top$. As we make use of an AR(1), $\boldsymbol{F}_t(u) = u_{t-1}$ and $\boldsymbol{F}_t(v) = v_{t-1}$. For all parameters in both models, we find strong prior-to-posterior learning. We take 10,000 samples from the posteriors of both the SSGP and Stan models. After normalizing priors and posteriors to $[0,1]$, Figure 4.2 shows slightly wider posteriors for the SSGP fit versus the output from Stan.

This is intuitive, since our SSGP is trained on only 50 model runs, whereas the Stan code evaluates the computer model at each iteration of the MCMC. The time used to fit both models was similar, indicating the SSGP could greatly reduce computational effort if model evaluations were instead expensive.

Model	RMSE	No. of ODE Solves
spatial (no bias)	0.301	50
spatial (with bias)	0.207	50
Stan ODE model	0.224	10000

This initial application illustrates that our methodology provides comparable performance to directly evaluating

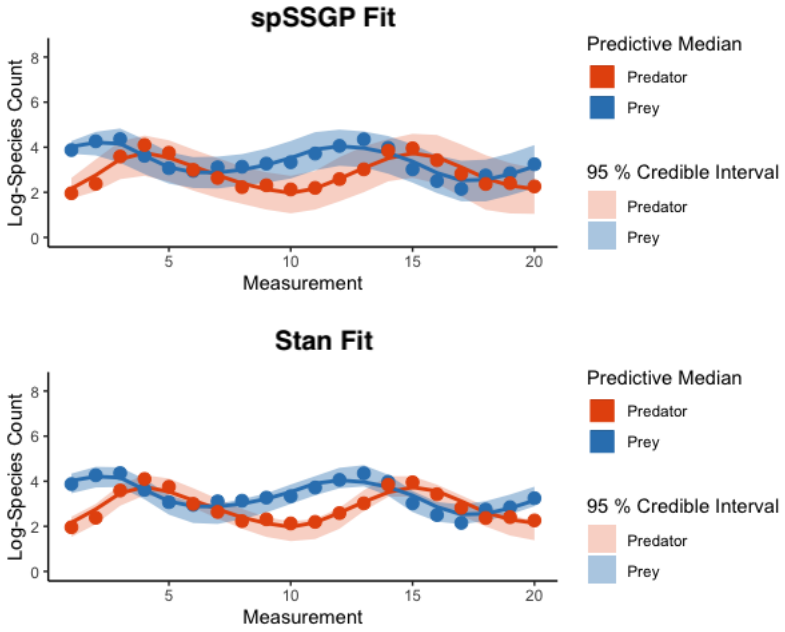


Figure 4.3: SSGP vs. Stan predictive distribution

the computer model, but in the following examples, the computer model may be too expensive to evaluate iteratively and thus necessitates the emulation/calibration methodology.

4.3.2 Nonlinear Partial Differential Equation Calibration

This section illustrates our methodology applied to a spatiotemporal computer model arising from a set of coupled nonlinear PDEs. PDEs are a natural way to describe dynamics of a continuous outcome across space and time. Though often technical, the dynamics generated through such systems can be highly desirable in modeling the natural world, see (Murray, 2003) for examples. Though such models are qualitatively advantageous, an ongoing area of research is enabling efficient parameter inference when connecting to real-world data. PDE models of applied interest almost never admit analytic solutions, and thus numerical approaches that involve discretizing and looping over the entire spatial domain are required. This is a computationally arduous task. Bayesian approaches to parameter inference have been investigated (Salter and Williamson, 2019), (Cockayne et al., 2017), (Wang et al., 2021) but often center on probabilistically modeling discretization error induced through the numerical solutions. If coarser meshes are used in the numerical solvers, often computational gains can be made at the expense of solution accuracy. In contrast, our methodology may utilize a few precise but

costly numerical evaluations where error becomes negligible. In this way, we reduce a source of uncertainty and focus on parameter calibration. Our methodology enables black-box inference of PDEs, where the only required input is a collection of numerical solutions across a set of parameter design points.

We model with the set of coupled PDEs shown in (4.23). This set of partial differential equations are popular within the applied mathematics community to capture qualitative behavior of spreading infection dynamics (Noble, 1974; Keeling and Rohani, 2008). Literature studying such systems of PDEs often focus on proving various mathematical properties, with no discussion about parameter inference or connecting to real-world data. Specifically, (Zhang and Wang, 2014) prove the existence of traveling waves in an influenza outbreak for a system such as (4.23). See Figure 4.5 for a visualization of this traveling wave phenomena when two seeding locations are used as initial sources of the disease. To motivate the system of equations under study, we modify a standard SIR compartment model by including a Laplacian term to induce diffusion of the populations over space. It is intuitive that in most circumstances, disease transmission is predominantly a localized process. For directly transmitted diseases, for example, transmission is most likely between individuals with the most intense interaction, which generally implies those in the same location. Additionally, movement of individuals between population centers facilitates the geographical spread of infectious diseases. This is written as

$$\begin{aligned}
\frac{\partial S_t(\boldsymbol{s})}{\partial t} &= -\eta_1 \frac{S_t(\boldsymbol{s}) I_t(\boldsymbol{s})}{N} + \alpha_1 \nabla^2 S_t(\boldsymbol{s}) \\
\frac{\partial I_t(\boldsymbol{s})}{\partial t} &= \eta_1 \frac{S_t(\boldsymbol{s}) I_t(\boldsymbol{s})}{N} - \eta_2 I_t(\boldsymbol{s}) + \alpha_2 \nabla^2 I_t(\boldsymbol{s}) \\
\frac{\partial R_t(\boldsymbol{s})}{\partial t} &= \eta_2 I_t(\boldsymbol{s}) + \alpha_3 \nabla^2 R_t(\boldsymbol{s}),
\end{aligned} \tag{4.23}$$

where $\nabla^2 f_t(\boldsymbol{s}) = \sum_{i=1}^{d} \frac{\partial^2 f_t(\boldsymbol{s})}{\partial \boldsymbol{s}_i^2}$. In applying our methodology, the computer model output is the numerical solutions for a fixed set of parameter settings, i.e, setting $\boldsymbol{x} = (\eta_1', \eta_2', \alpha_1', \alpha_2', \alpha_3')$ gives the outcome $y_t(\boldsymbol{s}, \boldsymbol{x}) = I_t(\boldsymbol{s})$. For our experiment, we fix in (4.23) $\alpha_1 = \alpha_3 = 0$ to induce the most heterogenous spatial dynamics by only allowing infections to diffuse across space. We add synthetic Gaussian error with unit variance to generate the field data $\boldsymbol{z}_{1:T}$ for a setting not in the computer model training set. The training and testing design space is shown below in Figure 4.4. We utilize the package `R` package ReacTran (Soetaert and Meysman, 2012) to provided high-quality numerical methods for solving complicated systems of differential equations. Across the design space of parameter values, we solve the PDE system on a 90×90 grid resulting in 8,100 spatial

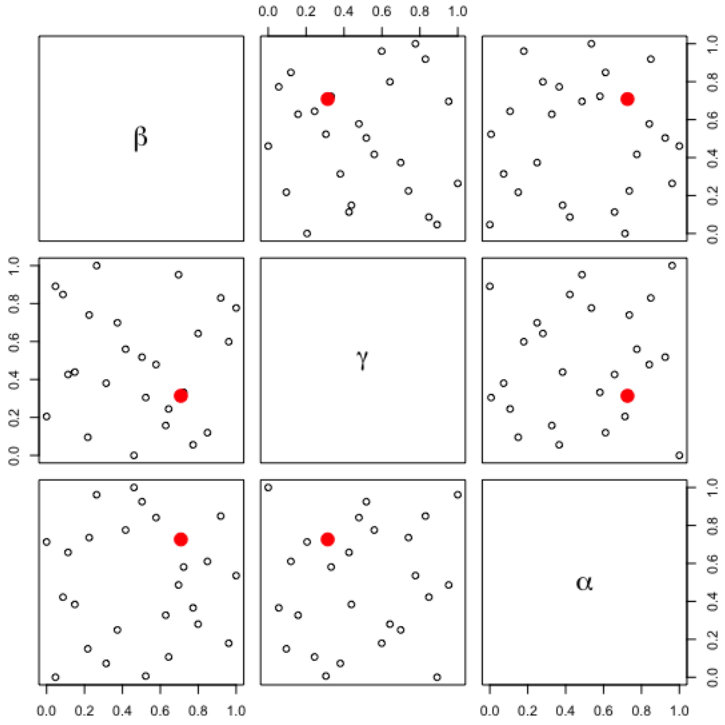


Figure 4.4: Training design space

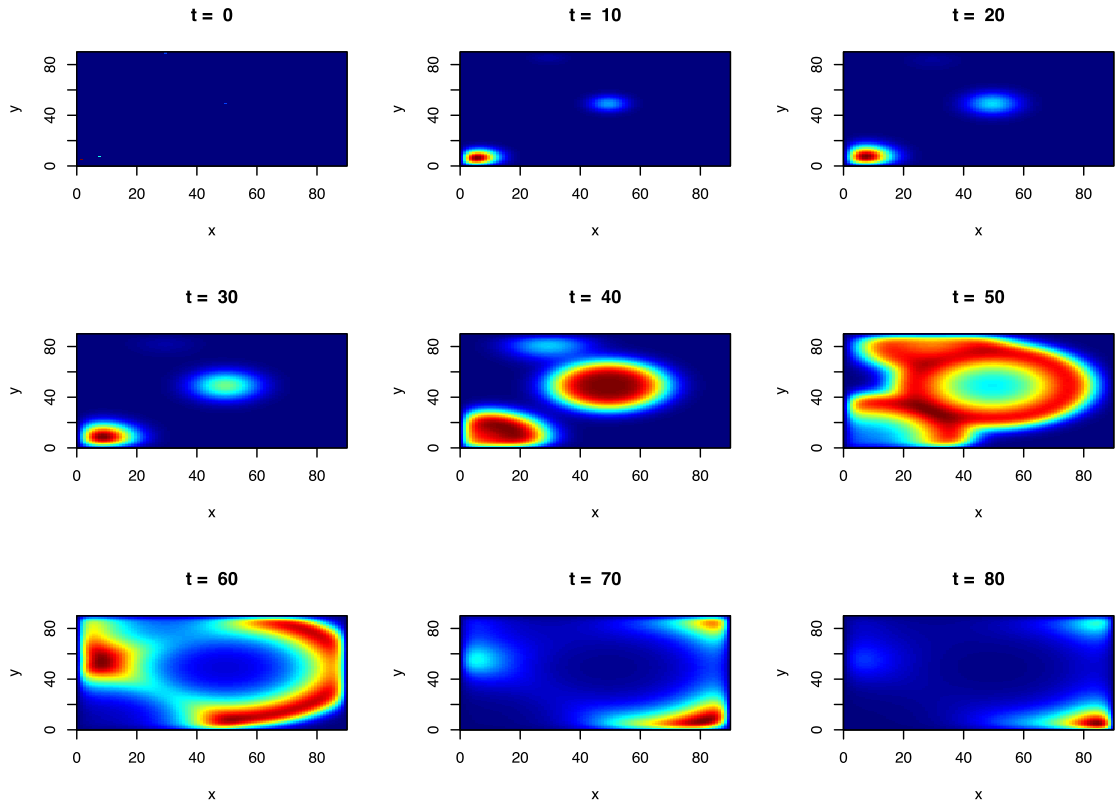


Figure 4.5: Spatiotemporal PDE dynamics

locations shown in Figure 4.5.

To reduce the dimensionality of the problem, we chose a set of 50 knot locations on a grid across the spatial domain. After training, we see below in Figure 4.6 strong Bayesian learning of calibration parameter values. This indicates that our SSGP model is successfully capturing the PDE system behavior and enabling efficient inference that circumvents the costly operation of repeatedly solving the system of equations on a large spatial domain.

Model	GRS	RMSE	No. of PDE Solves
heterogeneous model	-1107	2.32	25
spatial model	**-179**	**1.30**	**25**

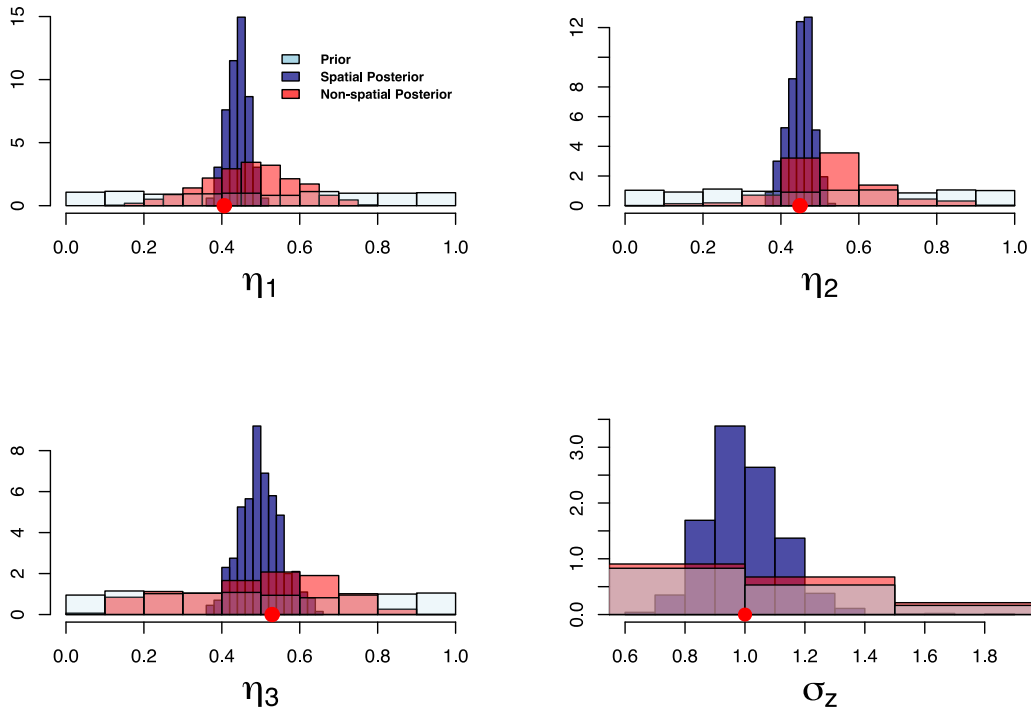


Figure 4.6: PDE Posteriors

4.3.3 Computers on Graphs and Networks

Network science studies how the macrostructure of interconnected systems such as telecommunication, economic, cognitive, or social networks affect global dynamics (Watts and Strogatz, 1998). Within a network, nodes represent discrete entities, and edges represent relations. Of prominent interest in applied modeling is understanding how global structure affects spread of a quantity throughout the system, termed network activation or network diffusion. To date, there exist various statistical software packages implementing diffusion or contagion processes across networks (Vega Yon and Valente, 2021; Siew, 2019). These computer simulations may be deterministic or stochastic in nature, though there exists a theoretical relation between the two (Mozgunov et al., 2018). We focus on calibrating a deterministic computer model popular within psychology and psycholinguistics communities (Chan and Vitevitch, 2009; Vitevitch et al., 2011). The `spreadr` package (Siew, 2019) implements this network diffusion simulation through, while discussion of parameter

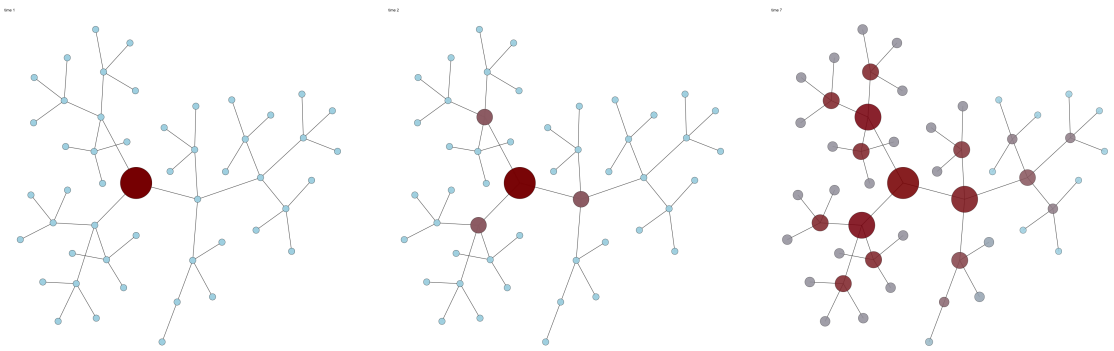


Figure 4.7: Network diffusion computer model

calibration is omitted. This justifies a use of our methodology.

We now define notation necessary for understanding the inputs to this computer simulation. For a network with n nodes, two parameters control the activation spread, known as *retention* and *decay*, denoted as r and d respectively.

- r - a scalar or a vector of length n that controls the proportion of activation retained by a node at each time step of the simulation.
- d - a scalar that controls the proportion of activation that is lost at each time step of the simulation.

There are three quantities of interest that define how activation spreads dynamically over time as functions of r and d: *reservoir*, *outflow*, and *inflow*. These are defined as follows:

- $\text{reservoir}(t,n) = r \times \text{inflow}(t,n)$.
- $\text{outflow}(t,n) = \dfrac{(1-d)(1-r) \times \text{inflow}(t,n)}{\deg(n)}$.
- $\text{inflow}(t,n) = \sum_{i=1}^{\deg(n)} \text{outflow}(t-1,n_i) + \text{reservoir}(t-1,n)$, where $\deg(n)$ denotes the number of connections to node n.

The computer simulation then computes these quantities for each node in the network, thereby algorithmically generating dynamics. A visualization of the spatiotemporal spreading dynamics is show in Figure 4.7.

Like in the previous section, we use a space-filling design to generate a small collection of plausible values for the parameters, after which we pick a random validation point not included in the training set. After 10,000 posterior draws, the prior-to-posterior learning is evident in Figure 4.7 and demonstrates our methodology is capable of calibrating a computer model generating dynamics across a network. Although it is possible to

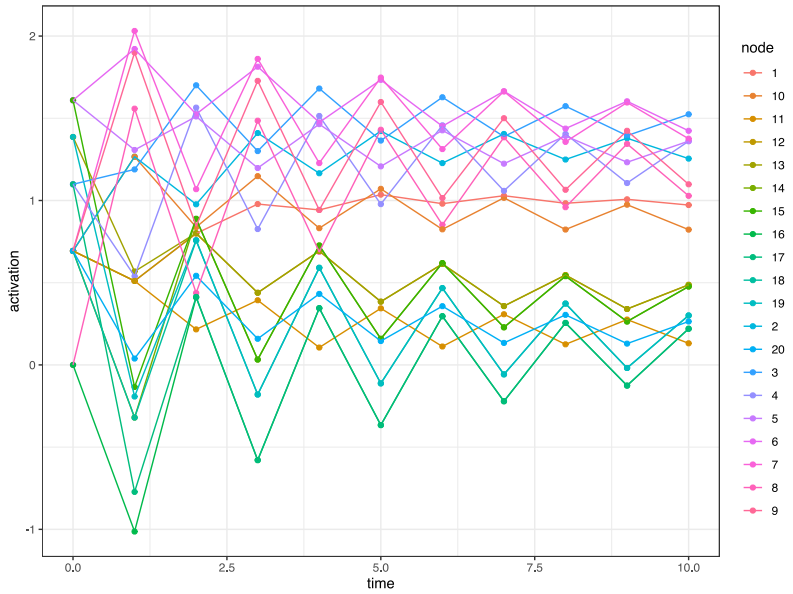


Figure 4.8: Individual node dynamics

define a Gaussian process over a graph, for this application we fix the correlation structure to arise from the adjacency matrix. As a symmetric matrix is positive definite if and only if its eigenvalues are all positive, we construct our matrix as

$$\boldsymbol{M} = \boldsymbol{I} + \rho \boldsymbol{A},$$

where $\boldsymbol{A}$ is the adjacency matrix. We choose ρ small enough to ensure that the correlation matrix is strictly diagonally dominant, meaning $|\rho| < 1/r$, where r is the valency of the regular graph. It follows that the eigenvalues of $\boldsymbol{C}$ are of the form $1 + \rho\lambda$ as λ ranges over the eigenvalues of $\boldsymbol{A}$. Valid graphs are those for which all eigenvalues of $\boldsymbol{A}$ satisfy the bound $\lambda > -1/\rho$. For a regular graph with valency r, all eigenvalues satisfy $|\lambda| \leq r$, which leads to the same sufficient condition $|\rho| < 1/r$ described above. We envision our method applicable beyond pure network activation models, for instance the forest fire cellular automata simulation models. Below we summarize the results of our two spatiotemporal model fits.

Model	GRS	RMSE	No. of computer model runs
heterogeneous model	1656	0.110	25
spatial model	**2142**	**0.095**	**25**

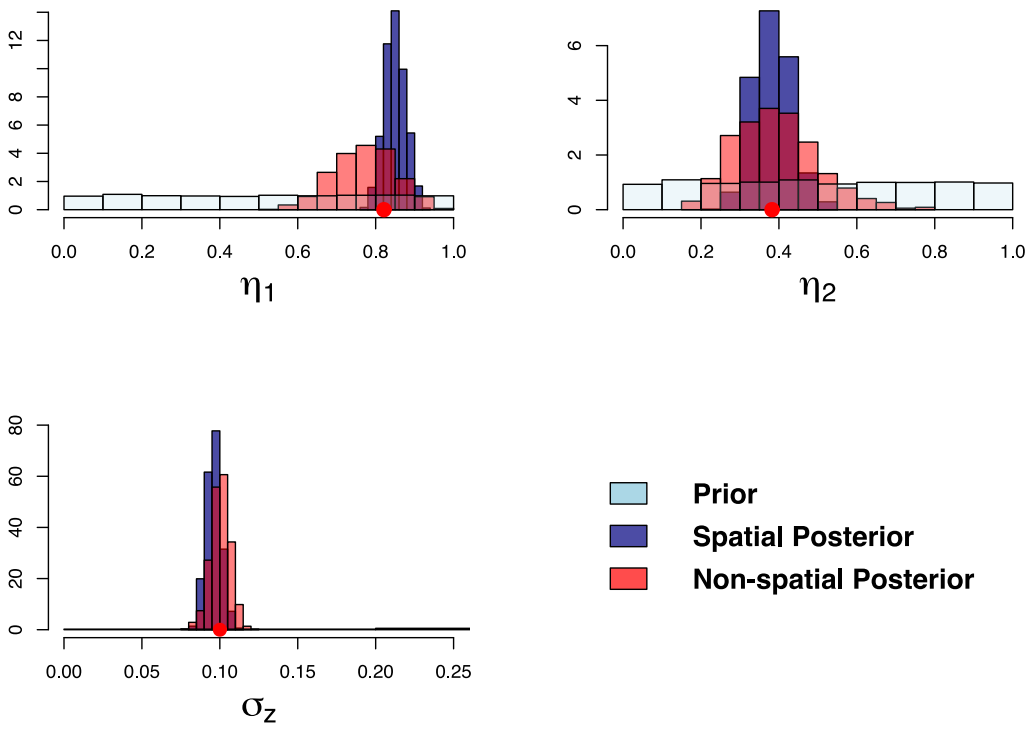


Figure 4.9: Network diffusion calibration posteriors

4.4 Discussion

In this work, we have developed a framework for the emulation and calibration of expensive spatiotemporal computer models. Our approach is broadly applicable to a variety of application areas to enable black-box calibration for the applied researcher. For potential further directions, Bayesian analysis of nonlinear PDEs is currently an active area of research (Wang et al., 2021). Making use of this state-space emulation/calibration approach for general classes of nonlinear PDEs past our example is likely fruitful. Additionally, the applicability of our model to network-like structures suggests that our methodology could also be applied for calibration of agent-based models evolving over space. Further scalability of our model by making use of spatial partitioning such as (Gramacy and Lee, 2007) or the newly-developed work of meshed Gaussian processes (Peruzzi et al., 2020). We have demonstrated a dynamic model bias term, and using reduced-rank Gaussian predictive processes allows us to scale our methodology to large space-time calibration problems. This dynamic spatiotemporal approach is likely to work well in calibrating computer models with low to medium-dimensional parameter input spaces. For high-dimensional computer simulations, inference of Gaussian process hyperparameters becomes cumbersome due to the Metropolis-Hastings steps required. Therefore, a sequential screening procedure may be employed to identify the most influential parameters (Welch et al., 1992; Merrill et al., 2021; Surer and Plumlee, 2021). There also exist opportunity to make use of generalized dynamic linear models to extend this state-space framework to computer outcomes that fall within the exponential family (West et al., 1985). Lastly, the burgeoning field of sequential Monte Carlo analysis is applicabled to build more general statistical emulators within this framework (Hirt and Dellaportas, 2019). We developed our methods in `C++` with functions available in `R` by way of the Rcpp package (Eddelbuettel and François, 2011).

4.5 Software and Computation

4.5.1 Forward-Filter-Backward-Sampling Computation

This appendix details the recursive computation necessary for Bayesian inference of the state-space model of (4.2). Consider a multivariate autoregressive state-space structure of the form

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{F}_t \boldsymbol{\phi}_t + \boldsymbol{\nu}_t, \quad \boldsymbol{\nu}_t \sim \mathcal{N}_N(0, v_t \boldsymbol{V}) \\ \boldsymbol{\phi}_t &= \boldsymbol{G}_t \boldsymbol{\phi}_{t-1} + \boldsymbol{\omega}_t, \quad \boldsymbol{\omega}_t \sim \mathcal{N}_{pS}(0, v_t \boldsymbol{W}). \end{aligned} \tag{4.24}$$

In applying (4.24) for spatiotemporal modeling, assume the outcome $\boldsymbol{y}_t$ is vectorized over space, so the correlation for measurement equation takes the form $\boldsymbol{V} = \boldsymbol{I}_S \otimes \widetilde{\boldsymbol{V}}$, where $\widetilde{\boldsymbol{V}} = \text{corr}(\boldsymbol{y}_t | \boldsymbol{\phi}_t, v_t)$. The observation transition matrix takes the form $\boldsymbol{F}_t = \text{Blk Diag}[\boldsymbol{F}_t(\boldsymbol{s}_1), \ldots, \boldsymbol{F}_t(\boldsymbol{s}_S)]$. From this, we may arrive at closed-form expressions necessary in the forward pass of FFBS. The Kalman filter calculates the moments of the filtering distribution $p(\boldsymbol{\phi}_t, v_t | \boldsymbol{y}_{1:t})$ by recursive application of Bayes' rule. Specifically, the one-step-ahead predictive distribution, marginal likelihood, and filtering distribution are recursively obtained by assuming first $(\boldsymbol{\phi}_0, v_0^{-1}) \sim \mathcal{NG}(n_0, d_0, \boldsymbol{m}_0, \boldsymbol{M}_0)$ and proceeding through time.

We now demonstrate the modular function calls involved in the implementation of Algorithm 2 and Algorithm 3. The implementation is written in `C++`, and functions are accessible to the `R` environment through `Rcpp`. The only requisite `Rcpp` libraries are RcppArmadillo: `https://CRAN.R-project.org/package=RcppArmadilloand` and

RcppDist: `https://cran.r-project.org/web/packages/RcppDist/index.html` .

Algorithm 4 Kalman filter computation

Input: Data $\boldsymbol{y}_{1:T}$, Kalman filter starting values n_0, d_0, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$, and discount factor δ_1

Correlation matrices $\boldsymbol{V}$, $\boldsymbol{W}$

Output: Filtering distribution parameters at time $t = 0, \ldots, T$

function KALMANFILTER($\boldsymbol{y}_{1:T}, n_0, d_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{V}, \boldsymbol{W}$)

  for $t = 1$ to T **do**

    # Compute prior distribution $p(\phi_t, v_t^{-1} | \boldsymbol{y}_{1:t-1}) \sim \mathcal{NG}(\boldsymbol{a}_t, \boldsymbol{A}_t, n_t^*, d_t^*)$:

    $\boldsymbol{a}_t \leftarrow \boldsymbol{G}_t \boldsymbol{m}_{t-1}$

    $\boldsymbol{A}_t \leftarrow \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{W}$

    Set $n_t^* \leftarrow \delta_1 n_{t-1}, d_t^* \leftarrow \delta_1 d_{t-1}$

    # Compute one-step-ahead forecast $p(\boldsymbol{y}_t | \boldsymbol{y}_{1:t-1}) \sim \mathcal{T}(\boldsymbol{q}_t, \boldsymbol{Q}_t \frac{d_t^*}{n_t^*}, 2n_t^*)$:

    $\boldsymbol{q}_t \leftarrow \boldsymbol{F}_t \boldsymbol{a}_t$

    # Woodbury identity to compute to Q_t^{-1}

    $\boldsymbol{Q}_t^{-1} \leftarrow \boldsymbol{V}^{-1} - \boldsymbol{V}^{-1} \boldsymbol{F}_t \left(\boldsymbol{A}_t^{-1} + \boldsymbol{F}_t^\top \boldsymbol{V}^{-1} \boldsymbol{F}_t \right)^{-1} \boldsymbol{F}_t^\top \boldsymbol{V}^{-1}$

    # Compute filtering distribution $p(\phi_t, v_t^{-1} | \boldsymbol{y}_{1:t}) \sim \mathcal{NG}(\boldsymbol{m}_t, \boldsymbol{M}_t, n_t, d_t)$:

    $\boldsymbol{m}_t \leftarrow \boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{F}_t \boldsymbol{Q}_t^{-1} (\boldsymbol{y}_t - \boldsymbol{q}_t)$

    $\boldsymbol{M}_t \leftarrow \boldsymbol{A}_t - \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{F}_t \boldsymbol{A}_t^\top$

    $n_t \leftarrow n_t^* + \frac{N}{2}$

    $d_t \leftarrow d_t^* + \frac{1}{2} (\boldsymbol{y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{y}_t - \boldsymbol{q}_t)$

  end for

  return $\{n_t, d_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^T$

end function

```
void kalman(const mat& y, const cube& F,
```

```
        const mat& V, const mat& W,
        const vec& m0, const mat& M0,
        mat& a, cube& A,
        mat& m, cube& M,
        vec& n, vec& d,
        const double n0, const double d0,
        const double delta_w,
        const double delta_v){
uword T = y.n_cols;
uword S = m0.n_elem;
a.col(0) =  m0.col(0);
A.slice(0) = M0 + W;
invV = inv(V);
mat q = F.slice(0) * a.col(0);
mat Qinv = invV - invV * F.slice(0)*
    inv(inv(A.slice(0)) + F.slice(0).t() * invV * F.slice(0)) *
    F.slice(0).t() * invV;
m.col(0) = a.col(0) + A.slice(0) *
    F.slice(0).t() * Qinv * (y.col(0) - q.col(0));
M.slice(0) = A.slice(0) - A.slice(0) *
    F.slice(0).t() * Qinv * F.slice(0) * A.slice(0);
// modify n and d for discount factor
n(0) = delta_v * n0 + 0.5 * y.n_rows;
d(0) = delta_v * d0 +
    0.5 * dot(y.col(0) - q.col(0), Qinv * (y.col(0) - q.col(0)));
for(uword t=1; t<T; t++){
    // compute prior
    a.col(t) =  m.col(t-1);
    A.slice(t) = M.slice(t-1) + W;
```

```
        // compute one-step-ahead predictive distribution moments
        q = F.slice(t) * a.col(t);
        Qinv = invV - invV * F.slice(t)*
            inv(inv(A.slice(t)) + F.slice(t).t() * invV * F.slice(t)) *
            F.slice(0).t() * invV;
        // compute filtering distribution moments
        m.col(t) = a.col(t) +
            A.slice(t) * F.slice(t).t() * Qinv * (y.col(t) - q);
        M.slice(t) = (A.slice(t) - A.slice(t) * F.slice(t).t() *
            Qinv * F.slice(t) * A.slice(t));
        n(t) = delta_v * n(t-1) + 0.5 * y.n_rows;
        d(t) = delta_v * d(t-1) +
            0.5 * dot(y.col(t) - q, Qinv * (y.col(t) - q));
    }
}
```

Now that the requisite moments have been obtained from the filtering process, the backward sampling can proceed. This backward sampling is a stochastic version of the Kalman smoothing process of Liu and West (2009). This obtains posterior samples from $p(\boldsymbol{\phi}_{0:T}, v_{0:T}|\boldsymbol{y}_{1:T})$.

Algorithm 5 Backward sample

Input: Filtering parameters from Algorithm 4

Output: Posterior sample from $p(\boldsymbol{\phi}_{0:T}, v_{0:T}|\boldsymbol{y}_{1:T})$

function BACKWARDSAMPLE($\{n_t, d_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^{T}$)

  Draw $v_T^{-1}|\boldsymbol{y}_{1:T} \sim \mathcal{G}(n_T, d_T)$

  Draw $\boldsymbol{\phi}_T|\boldsymbol{y}_{1:T} \sim \mathcal{N}(\boldsymbol{m}_T, v_T\boldsymbol{M}_T)$

  for $t = T-1$ to 0 **do**

    Draw $v_t^*|\boldsymbol{y}_{1:T} \sim \mathcal{G}((1-\delta_1)n_t, d_t)$

    $v_t^{-1}|\boldsymbol{y}_{1:T} \leftarrow \delta_1 v_{t+1}^{-1} + v_t^*$

    Compute $\boldsymbol{s}_t = \boldsymbol{m}_t + \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}(\boldsymbol{s}_{t+1} - \boldsymbol{a}_{t+1})$

$$\boldsymbol{S}_t = \boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}(\boldsymbol{A}_{t+1} - \boldsymbol{S}_{t+1})\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t$$

    Draw $\boldsymbol{\phi}_t|\boldsymbol{y}_{1:T} \sim \mathcal{N}(\boldsymbol{s}_t, v_t\boldsymbol{S}_t)$

  end for

  return $v_{0:T}, \boldsymbol{\phi}_{0:T}|\boldsymbol{y}_{1:T}$

end function

```
void backwardSample(mat& theta,
                    const mat& a,
                    const cube& A,
                    const mat& m,
                    const cube& M,
                    const vec v){
    uword T = a.n_cols;
    double vinv;
    uword T = n.n_elem;
    double vinv, v_star;
    vinv = Acpp::rgamma(1, n(T-1), 1.0 / d(T-1))(0);
```

```
    v(T-1) = 1.0 / vinv;
    mat s = m.col(T-1);
    mat S = M.slice(T-1);
    theta.col(T-1) = rmvnorm(1, s, v(T-1)*S).t();
    mat Ainv = R.slice(0);
    for(int t = T-2; t >= 0; t--){
        v_star = Rcpp::rgamma(1, (1-delta_v) * n(t+1), 1.0 / d(t))(0);
        vinv = delta_v * vinv + v_star;
        v(t) = 1.0 / vinv;
        Ainv = inv(R.slice(t+1));
        s = m.col(t) + M.slice(t)*Ainv*(s-a.col(t+1));
        S = M.slice(t) + M.slice(t)*Ainv*(R.slice(t+1)-S)*Ainv*M.slice(t);
        theta.col(t) = rmvnorm(1, s, v(t)*S)).t();
    }
}
```

CHAPTER 5

Discussion and Future Work

In this final chapter, we detail future steps towards building surrogate models capable of learning a wider range of nonlinear behavior through mixtures of state-space Gaussian process. This work builds upon multiprocess models and Markov-switching state-space frameworks, popular within the time series and econometrics literature (Kim and Nelson, 2017; Frühwirth-Schnatter, 2006). We present extensions towards new dynamic emulator methodology and an algorithm for sampling inspired through the work of Niemi and West (2010). Software implementation and application is left as future development. Finally, we conclude with a brief discussion on associated challenges and opportunities in modeling with mechanistic systems and expensive computer simulations.

5.1 Mixture State-Space Gaussian Processes

Methodology for our approach towards nonlinear surrogate modeling makes use of mixtures, state-space methods, and Gaussian process regression. We present the methodology tailored for inference via FFBS. The Bayesian multiprocess state-space model was initially introduced by Harrison and Stevens (1976) and involves a collection of state-space models that are posited to capture the dynamic data. The multiprocess framework is computationally demanding – especially in the case of Markov-switching mixture models (Kim and Nelson, 2017; Frühwirth-Schnatter, 2006) – but proves effective in identifying and adapting to structural changes with minimum loss of predictive and learning capacity. In the age of distributed and parallel computing, these frameworks are more computationally tractable than in the past. A concrete example is illustrated through the use of discount factors to specify a time-varying process on the variance components. Recall a discount learning specification for model (4.2.3) utilized either a beta-gamma process for the observational variance or modified expressions of the Kalman filter to induce time-varying but deterministic dynamics on the latent

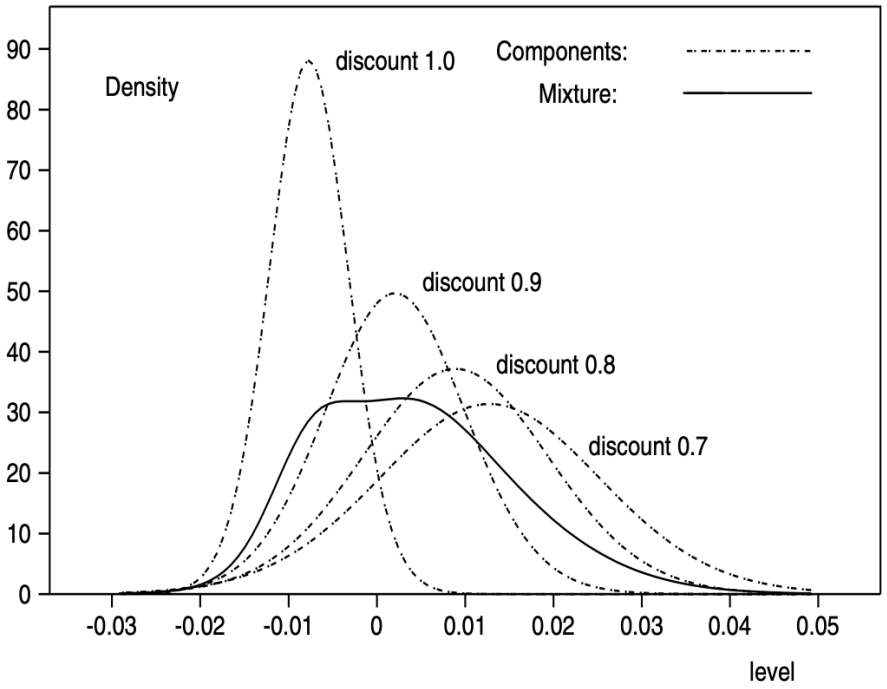


Figure 5.1: Mixture forecast for multiple state-space models (West and Harrison, 1997)

process (co)variance. These discount factors controlled the time dynamics and relative stability of the processes. Uncertainty about the discount factor can be incorporated into the modeling process by defining multiple state-space models, each with separate learning rates. For example, in Figure 5.1, we see a mixture estimate for a latent state μ_t has components significantly influenced by the discount learning rate. This mixture distribution is therefore a principled way to account for this uncertainty. Formally, consider K separate state-space models. To specify the mixture dynamic emulator, we follow the emulator construction of Chapter 4, while also introducing a latent indicator S. Conditioning on this latent indicator $S = k$, the multivariate time-varying autoregressive multiprocess state-space emulator is written,

$$\begin{aligned} y_t(\boldsymbol{x}) &= \boldsymbol{f}_t(\boldsymbol{x})^\top_{[k]}\boldsymbol{\phi}_t + \varepsilon_t(\boldsymbol{x}), \quad \varepsilon_t(\boldsymbol{\phi}) \sim \mathcal{GP}\left(0, C(\cdot;\boldsymbol{\beta}_{[k]})\right) \\ \boldsymbol{\phi}_t &= \boldsymbol{G}_t^{[k]}\boldsymbol{\phi}_{t-1} + \boldsymbol{\tau}_t, \quad \boldsymbol{\tau}_t \sim \mathcal{N}_p\left(0, \boldsymbol{W}_{[k]}\right), \end{aligned} \tag{5.1}$$

where $\boldsymbol{f}_t(\boldsymbol{x})_{[k]}$ is the $p \times 1$ autoregressive transition vector at input $\boldsymbol{x}$; $\boldsymbol{\phi}_t$ is the $p \times 1$ latent state; $G_t^{[k]}$ is the $p \times p$ state transition matrix; and $\boldsymbol{W}_{[k]}$ is the $p \times p$ matrix defined as $W_{[k]} := \text{cov}(\boldsymbol{\phi}_t)$ for the model k. Notice multiple Gaussian processes are incorporated into the model through this mixture framework. Conditional

on $S = k$ and for a realization across computer input space, the structure is of linear Gaussian multiprocess state-space form,

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{F}_t^{[k]}\boldsymbol{\phi}_t + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}_N\left(\mathbf{0}, \boldsymbol{V}_{[k]}\right) \\ \boldsymbol{\phi}_t &= \boldsymbol{G}_t^{[k]}\boldsymbol{\phi}_{t-1} + \boldsymbol{w}_t, \quad \boldsymbol{w}_t \sim \mathcal{N}_p\left(\mathbf{0}, \boldsymbol{W}_{[k]}\right). \end{aligned} \tag{5.2}$$

where $\boldsymbol{V}_{[k]}$ is the application of $\boldsymbol{C}(\cdot;\boldsymbol{\beta}_{[k]})$ to the computer input realization $\mathcal{X}$. Conditional on state $S = k$, the joint distribution necessary for emulator prediction then takes the form

$$\begin{bmatrix} y_t^*(\boldsymbol{\eta}) \\ \boldsymbol{y}_t \end{bmatrix} \Big| S = k, \boldsymbol{\eta}, \boldsymbol{\phi}_t, \boldsymbol{\beta}_{[k]} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f}_t(\boldsymbol{\eta})_{[k]}^\top \boldsymbol{\phi}_t \\ \boldsymbol{F}_t^{[k]}\boldsymbol{\phi}_t \end{bmatrix}, \begin{bmatrix} C(\boldsymbol{\eta};\boldsymbol{\beta}_{[k]}) & \boldsymbol{r}(\boldsymbol{\eta})_{[k]}^\top \\ \boldsymbol{r}(\boldsymbol{\eta})_{[k]} & \boldsymbol{V}_{[k]} \end{bmatrix} \right),$$

with $\boldsymbol{r}(\boldsymbol{\eta})_{[k]} = \left(\boldsymbol{C}(\boldsymbol{x}_1, \boldsymbol{\eta};\boldsymbol{\beta}_{[k]}), \ldots, \boldsymbol{C}(\boldsymbol{x}_N, \boldsymbol{\eta};\boldsymbol{\beta}_{[k]})\right)^\top$ a $N \times 1$ vector. Like in Section 4.2.1, the emulator predictive density at input $\boldsymbol{\eta}$ is given after conditioning so that

$$y_t^*(\boldsymbol{\eta}) \mid S, \boldsymbol{y}_t, \boldsymbol{\eta}, \boldsymbol{\phi}_t, \boldsymbol{\beta}_{[k]} \sim \mathcal{N}(\widetilde{\mu}_t(\boldsymbol{\eta}), \widetilde{\sigma}_t^2(\boldsymbol{\eta}))$$

where

$$\begin{aligned} \widetilde{\mu}_t(\boldsymbol{\eta}) &:= \boldsymbol{f}_t(\boldsymbol{\eta})_{[k]}^\top \boldsymbol{\phi}_t + \boldsymbol{r}(\boldsymbol{\eta})_{[k]}^\top \boldsymbol{V}_{[k]}^{-1}\left(\boldsymbol{y}_t - \boldsymbol{F}_t^{[k]}\boldsymbol{\phi}_t\right) \\ \widetilde{\sigma}_t^2(\boldsymbol{\eta}) &:= C(\boldsymbol{\eta};\boldsymbol{\beta}_{[k]}) - \boldsymbol{r}(\boldsymbol{\eta})_{[k]}^\top \boldsymbol{V}_{[k]}^{-1} \boldsymbol{r}(\boldsymbol{\eta})_{[k]}. \end{aligned} \tag{5.3}$$

Sampling and parameter inference for model (5.1) is developed in the next section. We conclude by mentioning Markov-switching state-space models are a further generalization enabling discrete time-varying dynamics of indicator S. The multiprocess model of this chapter may be considered a special case of the Markov-switching framework, where $S_t \equiv S$ for all t. In the context of building an emulator or surrogate model, the generalized switching paradigm allows us to enable time-varying shifts in mean and covariance structure of the Gaussian process, meaning it is no longer believed a single state-space model is appropriate for all t. This allows us to accommodate nonlinear computer model dynamics. For methodological textbook references on Markov mixtures, see (Frühwirth-Schnatter, 2006; Kim and Nelson, 2017). Initial building blocks towards implementation are developed through the Bayesian hidden Markov and blocked Gibbs sampler hosted at

`https://github.com/ian-frankenburg/HMMbayes`. The inferential feasibility of such switching is future work, but state-space regime switching models have successfully been applied in numerous econometric settings (Kim and Nelson, 2017), which suggests an opportunity for new application towards analysis of computer models and simulations.

5.1.1 Inference in Multiprocess State-Space Models

Unlike the filtering algorithms in Chapter 4, the mixture framework is more involved. This construction of a mixture state-space Gaussian process emulator is, in theory, an initial step in building a paradigm for calibration and uncertainty quantification of nonlinear, jumping, or stochastic computer models.

Posterior inference in this class of dynamic multiprocess models assumes a single state-space model holds for all time t. Inference is essentially an augmented version of the sampling in Chapter 4. Let S be a discrete indicator with support on set $\mathcal{S} = \{1, \ldots, K\}$ for some positive integer K. A probability mass function over the indicator space can be formed through $p_t(k) = \Pr(S = k \mid \boldsymbol{y}_{1:t})$. Denoting $l_t(k) = p(\boldsymbol{y}_{1:t} \mid S = k, \boldsymbol{y}_{1:t-1})$, we may update through $p_t(k) = c_t p_{t-1}(k) l_t(k)$, with the normalizing constant $c_t = \left(\sum_{k=1}^{K} p_{t-1}(k) l_t(k)\right)^{-1}$. The filtering distribution $p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t})$ is given by

$$p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t}) = \sum_{k=1}^{K} p(\boldsymbol{\phi}_t \mid S = k, \boldsymbol{y}_{1:t})\, p_t(k) \tag{5.4}$$

Our approach to sampling from $p(\boldsymbol{\phi}_t | \boldsymbol{y}_{1:T})$ is to augment the FFBS algorithm with an additional sampling step. In summary, for multiprocess Gaussian state-space models, if the hidden model indicator S takes on K values, the filter density is a mixture of K normal distributions:

$$p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_{1:t}) = \sum_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\phi}_t \middle| \boldsymbol{m}_t^{[k]}, \boldsymbol{M}_t^{[k]}\right) p_t(k), \tag{5.5}$$

where the number of components remains fixed for all $t = 1, \ldots, T$. The moments of the various components are obtained by running K parallel Kalman filters as shown in Algorithm 7. Each function call works by assuming that the state of S is equal to k, for $k = 1, \ldots, K$. Our filtering-smoothing algorithm is inspired through the work of (Niemi and West, 2010) and targets the posterior $p(\boldsymbol{\phi}_{0:T} | \boldsymbol{y}_{1:T}, \boldsymbol{V}_{[1,\ldots,K]}, \boldsymbol{W}_{[1,\ldots,K]})$ by a simple data

augmentation scheme shown in Algorithm 6. Recall the smoothing distribution

$$p(\boldsymbol{\phi}_{0:T} \mid \boldsymbol{y}_{1:T}) = \prod_{t=0}^{T} p(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1:T}, \boldsymbol{y}_{1:T}) \tag{5.6}$$

is computable through the DAG model structure to imply $p(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1:T}, \boldsymbol{y}_{1:T}) = p(\boldsymbol{\phi}_t \mid \boldsymbol{\phi}_{t+1}, \boldsymbol{y}_{1:t})$. This smoothing distribution provides the posterior of interest. For now, we assume known the Gaussian process hyperparameters of $\boldsymbol{\beta}$ and ψ within each mixture component to illustrate sampling from the posterior through the smoothing distributions

$$p(\boldsymbol{\phi}_t|\boldsymbol{\phi}_{t+1}, \boldsymbol{y}_{1:t}) = \sum_{k=1}^{K} q_t(k)\mathcal{N}(\boldsymbol{s}_t, \boldsymbol{S}_t), \tag{5.7}$$

where for $t = T-1, \ldots, 0$ $q_t(k) = p_t(k)\mathcal{N}\left(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{a}_{t+1}^{[k]}, \boldsymbol{R}_{t+1}^{[k]}\right) / \sum_{j=1}^{K} q_t(j)$. Algorithm 6 thus samples from the smoothing distribution (5.7).

5.2 Conclusion

This dissertation has provided examples and strategies for modeling with mechanistic structure and deterministic computer simulations. In Chapter 3, an inexpensive mechanistic model was embedded within an MCMC sampling routine and shown viable to calibrate from noisy epidemiological time series data. Chapter 4 developed a strategy for calibration when the computer model is expensive to evaluate and generates high-dimensional spatiotemporal data. There exist further opportunity to build even more scalable space-time surrogate functions with the recent literature on meshed Gaussian processes (Peruzzi et al., 2020). Additionally, state-space methods allow the analyst to incorporate mechanistic structure into the mean and covariance function of the Gaussian processes driving the stochastic innovations. The structural time series and engineering control literature are likely useful for building bespoke, customized emulators in the future (West and Harrison, 1997).

Algorithm 6 FFBS for mixture state-space model

Input: Kalman filter starting values $\boldsymbol{m}_0^{[1,\ldots,K]}$, $\boldsymbol{M}_0^{[1,\ldots,K]}$

Covariance matrices $\boldsymbol{V}_{[1,\ldots,K]}$, $\boldsymbol{W}_{[1,\ldots,K]}$

Output: Samples from mixture posterior and hidden indicator variable

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

  for $k = 1$ **to** K **do in parallel**

    # Run K Kalman filters in (5.1) in parallel

    $\{p_t^{[k]}, \boldsymbol{a}_t^{[k]}, \boldsymbol{A}_t^{[k]}, \boldsymbol{m}_t^{[k]}, \boldsymbol{M}_t^{[k]}\}_{t=0}^{T} \leftarrow \texttt{MultiKalmanFilter}(\boldsymbol{m}_0^{[k]}, \boldsymbol{M}_0^{[k]}, \boldsymbol{V}_{[k]}, \boldsymbol{W}_{[k]})$

    $\{q_t^{[k]}, \boldsymbol{\phi}_t^{[k]}\}_{t=0}^{T} \leftarrow \texttt{MultiSmooth}(\{\boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^{T})$

  end parallel for x

  This filtering distribution is: $p(\boldsymbol{\phi}_t \mid \boldsymbol{y}_t) = \sum_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\phi}_t \middle| \boldsymbol{s}_T^{[k]}, \boldsymbol{S}_T^{[k]}\right) \Pr(S = k \mid \boldsymbol{y}_t)$

  for t in 0 to T **do**

    Set $\boldsymbol{\pi} = (q_t(1), \ldots, q_t(K)) \,/\, \sum_{j=1}^{K} q_t(j)$.

    Draw $z_t \sim \text{Categorical}(\boldsymbol{\pi})$.

    Save $\boldsymbol{\phi}_t^{(i)} \leftarrow \boldsymbol{\phi}_t^{[z_t]}$

  end for

end for

5.3 Appendix: Supplemental algorithms

Algorithm 7 Filtering for multiprocess model

Input: Starting value p_0, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$

Covariance matrices $\boldsymbol{V}$, $\boldsymbol{W}$

Output: Filtering distribution parameters at time $t = 0, \ldots, T$

function MULTIKALMANFILTER($p_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{V}, \boldsymbol{W}$)

  for $t = 1$ to T **do**

    # Compute prior distribution $p(\boldsymbol{\phi}_t|\boldsymbol{y}_{1:t-1}) \sim \mathcal{N}(\boldsymbol{a}_t, \boldsymbol{A}_t)$:

    $\boldsymbol{a}_t \leftarrow \boldsymbol{G}_t \boldsymbol{m}_{t-1}$

    $\boldsymbol{A}_t \leftarrow \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{W}$

    # Compute one-step-ahead forecast $p(\boldsymbol{y}_t|\boldsymbol{y}_{1:t-1}) \sim \mathcal{N}(\boldsymbol{q}_t, \boldsymbol{Q}_t)$:

    $\boldsymbol{q}_t \leftarrow \boldsymbol{F}_t \boldsymbol{a}_t$

    $\boldsymbol{Q}_t \leftarrow \boldsymbol{F}_t \boldsymbol{A}_t \boldsymbol{F}_t^\top + \boldsymbol{V}$

    # Compute filtering distribution $p(\boldsymbol{\phi}_t|\boldsymbol{y}_{1:t}) \sim \mathcal{N}(\boldsymbol{m}_t, \boldsymbol{M}_t)$:

    $\boldsymbol{m}_t \leftarrow \boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{F}_t \boldsymbol{Q}_t^{-1} (\boldsymbol{y}_t - \boldsymbol{q}_t)$

    $\boldsymbol{M}_t \leftarrow \boldsymbol{A}_t - \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{F}_t \boldsymbol{A}_t^\top$

    # Define $p_t := \Pr(S = k \mid \boldsymbol{y}_{1:t-1})$

    $p_t \leftarrow p_{t-1} \mathcal{N}(\boldsymbol{y}_t \mid \boldsymbol{q}_t, \boldsymbol{Q}_t)$

  end for

  return $\{p_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^T$

end function

The smoothing distributions for the mixture state-space model are presented below.

Algorithm 8 Smoothing for multiprocess model

function MULTISMOOTH($\{p_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^{T}$)
  Set $\boldsymbol{s}_T \leftarrow \boldsymbol{m}_T$ and $\boldsymbol{S}_T \leftarrow \boldsymbol{M}_T$
  Draw $\boldsymbol{\phi}_T \sim \mathcal{N}(\boldsymbol{s}_T, \boldsymbol{S}_T)$
  Set $q_T \leftarrow p_T$
  for $t = T-1$ to 0 **do**
    $\boldsymbol{s}_t \leftarrow \boldsymbol{m}_t + \boldsymbol{M}_t \boldsymbol{G}_{t+1}^{\top} \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{s}_{t+1} - \boldsymbol{a}_{t+1})$
    $\boldsymbol{S}_t \leftarrow \boldsymbol{M}_t - \boldsymbol{M}_t \boldsymbol{G}_{t+1}^{\top} \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{A}_{t+1} - \boldsymbol{S}_{t+1}) \boldsymbol{A}_{t+1}^{-1} \boldsymbol{G}_{t+1} \boldsymbol{M}_t$
    Draw $\boldsymbol{\phi}_t \sim \mathcal{N}(\boldsymbol{s}_t, \boldsymbol{S}_t)$
    $q_t \leftarrow p_t \mathcal{N}(\boldsymbol{\phi}_{t+1} \mid \boldsymbol{a}_{t+1}, \boldsymbol{A}_{t+1})$
  end for
  return $\{q_t, \boldsymbol{\phi}_t\}_{t=0}^{T}$
end function

Bibliography

Baker, R., Peña, J.-M., Jayamohan, J., and Jérusalem, A. (2018). Mechanistic models versus machine learning, a fight worth fighting for the biological community? *Biology Letters*, 14:20170660.

Banerjee, S. (2017). High-Dimensional Bayesian Geostatistics. *Bayesian Analysis*, 12(2):583 – 614.

Banerjee, S., Carlin, B., and Gelfand, A. (2014). *Hierarchical Modeling and Analysis for Spatial Data*.

Banerjee, S., Gelfand, A., Finley, A., and Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 70:825–848.

Bayarri, M. J., Berger, J. O., and Liu, F. (2009). Modularization in Bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, 4(1):119 – 150.

Betancourt, M. (2018). A conceptual introduction to hamiltonian monte carlo.

Bonaccorsi, G., Pierri, F., Cinelli, M., Flori, A., Galeazzi, A., Porcelli, F., Schmidt, A. L., Valensise, C. M., Scala, A., Quattrociocchi, W., and Pammolli, F. (2020). Economic and social consequences of human mobility restrictions under covid-19. *Proceedings of the National Academy of Sciences*, 117(27):15530–15535.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan : A probabilistic programming language. *Journal of Statistical Software*, 76.

Carter, C. and Kohn, R. (1994). On gibbs sampling for state space models. *Biometrika*, 81.

Chan, K. Y. and Vitevitch, M. S. (2009). The influence of the phonological neighborhood clustering coefficient on spoken word recognition. *Journal of experimental psychology. Human perception and performance*, 35 6:1934–49.

Chatzilena, A., van Leeuwen, E., Ratmann, O., Baguelin, M., and Demiris, N. (2019). Contemporary statistical inference for infectious disease models using stan. *Epidemics*, 29:100367.

Cockayne, J., Oates, C., Sullivan, T., and Girolami, M. (2017). Probabilistic numerical methods for pde-constrained bayesian inverse problems. *AIP Conference Proceedings*, 1853(1):060001.

Cutler, D. M. and Summers, L. H. (2020). The covid-19 pandemic and the $16 trillion virus. *JAMA*, 324(15):1495–1496.

Diekmann, O., Heesterbeek, J., and Roberts, M. (2009). The construction of next-generation matrices for compartmental epidemic models. *Journal of the Royal Society, Interface / the Royal Society*, 7:873–85.

Dietz, K. and Heesterbeek, J. (2002). Daniel bernoulli's epidemiological model revisited. *Mathematical biosciences*, 180:1–21.

Dukic, V., Lopes, H., and Polson, N. (2012). Tracking epidemics with google flu trends data and a state-space seir model. *Journal of The American Statistical Association*, 107.

Eddelbuettel, D. and François, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18.

Eleftheriadis, S., Nicholson, T., Deisenroth, M., and Hensman, J. (2017). Identification of gaussian process state space models.

Farah, M., Birrell, P., Conti, S., and Angelis, D. D. (2014). Bayesian emulation and calibration of a dynamic epidemic model for a/h1n1 influenza. *Journal of the American Statistical Association*, 109(508):1398–1411.

Finley, A., Banerjee, S., and Gelfand, A. (2012). Bayesian dynamic modeling for large space–time datasets using gaussian predictive processes. *Journal of Geographical Systems*, 14:29–47.

Finley, A., Sang, H., Banerjee, S., and Gelfand, A. (2009). Improving the performance of predictive process modeling for large datasets. *Computational statistics & data analysis*, 53:2873–2884.

Frankenburg, I. and Banerjee, S. (2022). A compartment model of human mobility and early covid-19 dynamics in nyc. *The New England Journal of Statistics in Data Science*, pages 1–12.

Frühwirth-Schnatter, S. (2006). *Finite Mixture and Markov Switching Models*. Springer Series in Statistics. Springer New York.

Gandhi, M., Yokoe, D., and Havlir, D. (2020). Asymptomatic transmission, the achilles' heel of current strategies to control covid-19. *New England Journal of Medicine*, 382.

Garnett, R. (2022). *Bayesian Optimization*. Cambridge University Press. in preparation.

Gelfand, A., Diggle, P., Fuentes, M., and Guttorp, P. (2010). Handbook of spatial statistics. *Handbook of Spatial Statistics*.

Gelman, A. (2004). Parameterization and bayesian modeling. *Journal of the American Statistical Association*, 99.

Gelman, A. and Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457 – 472.

Gneiting, T. and Raftery, A. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378.

Gramacy, R. and Lee, H. (2007). Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103.

Gu, Y. (2020). Covid-19 projections using machine learning.

Hao, X., Cheng, S., Wu, D., Wu, T., Lin, X., and Wang, C. (2020). Reconstruction of the full transmission dynamics of covid-19 in wuhan. *Nature*, 584:1–7.

Harrison, P. J. and Stevens, C. F. (1976). Bayesian forecasting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 38(3):205–247.

Heffernan, J., Smith, R., and Wahl, L. (2005). Perspectives on the basic reproductive ratio. *Journal of the Royal Society, Interface / the Royal Society*, 2:281–93.

Hefley, T., Hooten, M., Russell, R., Walsh, D., and Powell, J. (2017). When mechanism matters: Bayesian forecasting using models of ecological diffusion. *Ecology Letters*, 20.

Hewitt, G. (1917). Conservation of wild life in canada. *Nature*, 99.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103:570–583.

Higdon, D., Kennedy, M., Cavendish, J., Cafeo, J., and Ryne, R. (2004). Combining field data and computer simulations for calibration and prediction. *SIAM J. Scientific Computing*, 26:448–466.

Hirt, M. and Dellaportas, P. (2019). Scalable bayesian learning for state space models using variational inference with smc samplers. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 76–86. PMLR.

Hoffman, M. and Gelman, A. (2011). The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15.

Hooten, M., Garlick, M., and Powell, J. (2013). Computationally efficient statistical differential equation modeling using homogenization. *Journal of Agricultural, Biological, and Environmental Statistics*, 18.

Ives, A. R. and Bozzuto, C. (2020). State-by-state estimates of r0 at the start of covid-19 outbreaks in the usa. *medRxiv*.

Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*.

Keeling, M. J. and Rohani, P. (2008). *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press.

Kennedy, M. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B*, 63:425–464.

Kermack, W. O. and McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences*, 115(772):700–721.

Kim, C.-J. and Nelson, C. (2017). *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*.

Kobayashi, G., Sugasawa, S., Tamae, H., and Ozu, T. (2020). Predicting intervention effect for covid-19 in japan: state space modeling approach. *BioScience Trends*, 14.

Kolmogorov, A. N. (1941). *Stationary Sequences in Hilbert Space*. Moscow University Mathematics Bulletin.

Lauer, S., Grantz, K., Bi, Q., Jones, F., Zheng, Q., Meredith, H., Azman, A., Reich, N., and Lessler, J. (2020). The incubation period of coronavirus disease 2019 (covid-19) from publicly reported confirmed cases: Estimation and application. *Annals of internal medicine*, 172.

Liu, F. and West, M. (2009). A Dynamic Modelling Strategy for Bayesian Computer Model Emulation. *Bayesian Analysis*, 4(2):393 – 411.

Liu, J. and Wu, Y. (1999). Parameter expansion for data augmentation. *Journal of the American Statistical Association*, 94.

Madjid, M., Safavi-Naeini, P., Solomon, S., and Vardeny, O. (2020). Potential effects of coronaviruses on the cardiovascular system: A review. *JAMA Cardiology*, 5.

Merrill, E., Fern, A., Fern, X., and Dolatnia, N. (2021). An empirical study of bayesian optimization: Acquisition versus partition. *Journal of Machine Learning Research*, 22(4):1–25.

Mozgunov, P., Beccuti, M., Horvath, A., Jaki, T., Sirovich, R., and Bibbona, E. (2018). A review of the deterministic and diffusion approximations for stochastic chemical reaction networks. *Reaction Kinetics, Mechanisms and Catalysis*, 123.

Murray, J. D. (2003). *Mathematical Biology II: Spatial Models and Biomedical Applications*, volume 18 of *Interdisciplinary Applied Mathematics*. Springer New York.

Neal, R. (2012). Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*.

Niemi, J. and West, M. (2010). Adaptive mixture modeling metropolis methods for bayesian analysis of nonlinear state-space models. *Journal of Computational and Graphical Statistics*, 19:260 – 280.

Noble, J. (1974). Geographic and temporal development of plagues. *Nature*, 250(5469):726—729.

Oakley, J., Hagan, A., and O'Hagan, A. (2004). Probabilistic sensitivity analysis of complex models: A bayesian approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66:751 – 769.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext / Universitext. Springer.

Osthus, D., Hickmann, K., Caragea, P., Higdon, D., and Del Valle, S. (2017). Forecasting seasonal influenza with a state-space sir model. *The Annals of Applied Statistics*, 11:202–224.

Peruzzi, M., Banerjee, S., and Finley, A. (2020). Highly scalable bayesian geostatistical modeling via meshed gaussian processes on partitioned domains. *Journal of the American Statistical Association*, pages 1–31.

Petris, G., Petrone, S., and Campagnoli, P. (2009). *Dynamic Linear Models with R*, volume 38.

Plummer, M. (2003). Jags: A program for analysis of bayesian graphical models using gibbs sampling.

Poole, D. and Raftery, A. E. (2000). Inference for deterministic simulation models: The bayesian melding approach. *Journal of the American Statistical Association*, 95(452):1244–1255.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(65):1939–1959.

Salter, J. M. and Williamson, D. B. (2019). Efficient calibration for high-dimensional computer model output using basis methods.

Santner, T., Williams, B., and Notz, W. (2003). *The Design and Analysis Computer Experiments*.

Scott, S. L. (2002). Bayesian methods for hidden markov models. *Journal of the American Statistical Association*, 97(457):337–351.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Shiode, N., Shiode, S., Rod-Thatcher, E., Rana, S., and Vinten-Johansen, P. (2015). The mortality rates and the space-time patterns of john snow's cholera epidemic map. *International Journal of Health Geographics*, 14:21.

Siew, C. (2019). spreadr: An r package to simulate spreading activation in a network. *Behavior Research Methods*, 51.

Soetaert, K. and Meysman, F. (2012). Reactive transport in aquatic ecosystems: Rapid model prototyping in the open source software r. *Environmental Modelling & Software*, 32:49–60.

Stan Development Team (2020). *Stan Modeling Language Users Guide and Reference Manual, Version 2.21.0*.

Stan Development Team (2021). RStan: the R interface to Stan. R package version 2.21.3.

Struthers, A. and Potter, M. (2019). *Differential Equations: For Scientists and Engineers*.

Surer, O. and Plumlee, M. (2021). Calibration using emulation of filtered simulation results. In *2021 Winter Simulation Conference (WSC)*, pages 1–12.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*.

Särkkä, S. and Solin, A. (2019). *Applied Stochastic Differential Equations*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.

Treiber, M. and Kanagaraj, V. (2015). Comparing numerical integration schemes for time-continuous car-following models. *Physica A: Statistical Mechanics and its Applications*, 419:183–195.

Turner, R., Deisenroth, M., and Rasmussen, C. (2010). State-space inference and learning with gaussian processes. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 868–875.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In Escalante, H. J. and Hofmann, K., editors, *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 3–26. PMLR.

U.S. Census Bureau (2019). 2010 census. U.S. Department of Commerce.

Vega Yon, G. and Valente, T. (2021). *netdiffuseR: Analysis of Diffusion and Contagion Processes on Networks*. R package version 1.22.3.

Vitevitch, M., Ercal, G., and Adagarla, B. (2011). Simulating retrieval from a highly clustered network: Implications for spoken word recognition. *Frontiers in psychology*, 2:369.

Wang, C., Liu, L., Hao, X., Guo, H., Wang, Q., Huang, J., He, N., Yu, H., Lin, X., Pan, A., Wei, S., and Wu, T. (2020a). Evolving epidemiology and impact of non-pharmaceutical interventions on the outbreak of coronavirus disease 2019 in wuhan, china.

Wang, J., Cockayne, J., Chkrebtii, O., Sullivan, T., and Oates, C. (2021). Bayesian numerical methods for nonlinear partial differential equations. *Statistics and Computing*, 31.

Wang, L., Zhou, Y., He, J., Zhu, B., Wang, F., Tang, L., Eisenberg, M., and Song, P. X. (2020b). An epidemiological forecast model and software assessing interventions on covid-19 epidemic in china. *medRxiv*.

Wang, W., Ding, Z., and Gui, Z. (2018). A stochastic differential equation sis epidemic model incorporating ornstein–uhlenbeck process. *Physica A: Statistical Mechanics and its Applications*, 509.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992). Screening, predicting, and computer experiments. *Technometrics*, 34:15–25.

West, M. and Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models (2nd Ed.)*. Springer-Verlag, Berlin, Heidelberg.

West, M., Harrison, P. J., and Migon, H. S. (1985). Dynamic generalized linear models and bayesian forecasting. *Journal of the American Statistical Association*, 80(389):73–83.

Wiener, N. (1964). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press.

Wikle, C. and Hooten, M. (2010). A general science-based framework for dynamical spatio-temporal models. *TEST*, 19:417–451.

Wikle, C., Zammit-Mangion, A., and Cressie, N. (2019). *Spatio-Temporal Statistics with R*.

Wooldridge, J. (2002). Econometric analysis of cross section and panel data. *Booksgooglecom*, 58.

Zellner, A. and Ando, T. (2010). Bayesian and non-bayesian analysis of the seemingly unrelated regression model with student- errors, and its application for forecasting. *International Journal of Forecasting*, 26:413–434.

Zhang, T. and Wang, W. (2014). Existence of traveling wave solutions for influenza model with treatment. *Journal of Mathematical Analysis and Applications*, 419:469–495.