Journal of Machine Learning Research 26 (2025) 1-43 Submitted 8/22; Revised 7/25; Published 7/25

Dynamic Bayesian Learning for Spatiotemporal Mechanistic Models

Sudipto Banerjee SUDIPTO@UCLA.EDU
Department of Biostatistics
University of California, Los Angeles
Los Angeles, CA 90025, USA

Xiang Chen XIANGCHEN@UCLA.EDU
Department of Biostatistics
University of California, Los Angeles
Los Angeles, CA 90025, USA

Ian Frankenburg IAN.FRANKENBURG@UCLA.EDU
Department of Biostatistics
University of California, Los Angeles
Los Angeles, CA 90025, USA

Daniel Zhou HUNANZHOU@G.UCLA.EDU
Department of Biostatistics
University of California, Los Angeles
Los Angeles, CA 90025, USA

Editor: Debdeep Pati

Abstract

We develop an approach for Bayesian learning of spatiotemporal dynamical mechanistic models. Such learning consists of statistical emulation of the mechanistic system that can efficiently interpolate the output of the system from arbitrary inputs. The emulated learner can then be used to train the system from noisy data achieved by melding information from observed data with the emulated mechanistic system. This joint melding of mechanistic systems employ hierarchical state-space models with Gaussian process regression. Assuming the dynamical system is controlled by a finite collection of inputs, Gaussian process regression learns the effect of these parameters through a number of training runs, driving the stochastic innovations of the spatiotemporal state-space component. This enables efficient modeling of the dynamics over space and time. This article details exact inference with analytically accessible posterior distributions in hierarchical matrix-variate Normal and Wishart models in designing the emulator. This step obviates expensive iterative algorithms such as Markov chain Monte Carlo or variational approximations. We also show how emulation is applicable to large-scale emulation by designing a dynamic Bayesian transfer learning framework. Inference on $\boldsymbol{\eta}$ proceeds using Markov chain Monte Carlo as a post-emulation step using the emulator as a regression component. We demonstrate this framework through solving inverse problems arising in the analysis of ordinary and partial nonlinear differential equations and, in addition, to a black-box computer model generating spatiotemporal dynamics across a graphical model.

Keywords: Bayesian melding, computer models, state-space models, Bayesian transfer learning, Gaussian process regression, mechanistic systems, spatiotemporal analysis, uncertainty quantification.

1. Introduction

Probabilistic learning of spatial-temporal processes governed by physical or mechanistic systems is of interest in a variety of scientific disciplines. Such models are sometimes represented through a system of equations derived from physical or scientific principles, or they may be arbitrarily complex computer programs that simulate a phenomenon. These models are broadly referred to as *computer models*[1] and, at least in the scope of this paper, are deterministic in the sense that rerunning the model with a specific set of inputs always produce identical value of the outputs. In what is considered a seminal manuscript in statistical science, Sacks et al. (1989) describes a framework for modeling the output as a realization of a stochastic process and laying the statistical foundations for designing computer experiments for efficient prediction. This field has blossomed into a substantial area within machine learning and statistical science with the incorporation of ideas from signal processing, dynamical systems, inverse problems, functional data analysis, non-parametric models, Gaussian processes, spatial statistics and several other fields (while a comprehensive review is not the aim of this article, we refer to Santner et al., 2019; Fang et al., 2005; Kang and Deng, 2020, and references therein for diverse perspectives).

Analyzing data from computer experiments refer to interpolating or predicting the output at new inputs after training the model using runs of the computer model without assuming mathematical tractability of the computer model. This is referred to as *emulation* (see Sacks et al., 1989; Conti and O'Hagan, 2010, with the latter offering an excellent perspective of relative recent literature) and comprises the construction of a statistical model that mimics the behavior of the computer model. Rather than assuming an explicit functional relationship between the inputs and outputs, a Gaussian process is used as a prior on the unknown function (see, e.g., O'Hagan, 1992; Haylock and O'Hagan, 1996; Rasmussen and Williams, 2005) and the outputs are assumed to be a realization of this process.

A second problem of interest is to learn about the parameters in the mechanistic system from field observations while also accounting for the information available from runs of the computer experiment. This is related to the problem of *calibration* using field data (see, e.g., Kennedy and O'Hagan, 2001; Oakley and O'Hagan, 2004; Higdon et al., 2004; Bayarri et al., 2007a; Higdon et al., 2008; Bayarri et al., 2009, for calibration using field observations in diverse contexts) and can be regarded as an inverse problem seeking optimal values of unknown parameters to be learned from observational field data. We depart somewhat from the paradigm of traditional calibration, where an "optimal" (perhaps unknown) value of the input is assumed. Instead pursue *melding* (Poole and Raftery, 2000; Gel et al., 2004) and synthesize information from mechanistic systems with field observations. Emulation and calibration becomes challenging when the computer model or posited functional relationship

1. In the rest of the article, we occasionally refer to the mechanistic system which we seek to model, learn, and estimate parameters from field data as a computer model to be consistent with terminology in statistics for referring to deterministic models.

is complex in nature or expensive to compute and any practicable learning framework must account for the scale of the problem.

We focus upon emulating and inferring on spatial-temporal mechanistic systems using a Bayesian hierarchical modeling framework. Figure 1 depicts emulation and calibration in our melding context. A hierarchical description of a physical process consists of a layered approach, whereby simpler conditional dependence structures specify complex relationships. Probabilistic learning proceeds from the posterior distribution given by

$$\begin{aligned}[\text{process, parameters} \mid \text{data}] \propto [\text{data} \mid \text{process}, \text{parameters}] \\ \times [\text{process} \mid \text{parameters}] \times [\text{parameters}] \, .\end{aligned} \tag{1}$$

At the top level, the probability model specifies the distribution of the data conditional on the physical process and any other parameters needed to describe the data generating mechanism. The next level represents the underlying mechanistic system as a realization of a stochastic process capturing physical or mechanistic knowledge (as promulgated by Sacks et al., 1989). The last level models uncertainty about parameters.

Probabilistic machine learning using (1) adopts Bayesian inference of all unknown quantities given the data available to the modeler. Bayesian inference for deterministic systems has been developed and explored in settings that resemble computer experiment settings. Examples, by no means exhaustive, include Bayesian melding developed by Poole and Raftery (2000) that pursues inference from deterministic simulation methods and has witnessed substantial use in climate modeling (see Gel et al., 2004, for one application in forecasting problems), Bayesian data assimilation (Wikle and Berliner, 2007) and Bayesian state space models formed by finite difference approximations of differential equations (see, e.g., Wikle, 2003; Stroud et al., 2010; Wikle and Hooten, 2010; Abdalla et al., 2020, for applications encompassing ecology, climate and industrial hygiene).

In this manuscript we specifically focus upon spatial-temporal mechanistic systems and builds upon a rapidly evolving literature in Bayesian learning for computer models (see Liu and West, 2009; Farah et al., 2014; Conti and O'Hagan, 2010; Gu and Berger, 2016; Gu et al., 2019; Gu and Shen, 2020; Gu and Li, 2022; Gu, 2022, and references therein). Notably, Gu et al. (2019) discusses software development and fast implementation of the methodology in Gu and Berger (2016) using experiments and field observations for vector-valued outputs while also focusing upon accelerating computations. Other work on combining Gaussian processes and state-space models may be found in the machine learning literature (see, e.g., Turner et al., 2010; Eleftheriadis et al., 2017), where the process is used to model the transition function rather than stochastic innovations.

Following Conti and O'Hagan (2010) and Gu and Shen (2020), we use matrix-variate distributions with rows and columns corresponding to inputs and locations, but we incorporate temporal emulation over discrete epochs leading to matrix-variate Bayesian dynamic models. In this regard, we differ from Gu and Shen (2020) who use continuous time Kalman filters. While sacrificing some richness in statistical inference, we aim to harness exact analytically tractable distribution theory to generate exact posterior samples without resorting to iterative algorithms such as Markov chain Monte Carlo (MCMC, Robert and Casella, 2005; Gamerman and Lopes, 2006), Variational Bayes (Ren et al., 2011; Blei et al., 2017) or Laplace approximations (Rue et al., 2009). We devise a matrix-variate Forward Filter Backward Sampling (FFBS) algorithm (Carter and Kohn, 1994; Frühwirth-Schnatter, 2006) to emulate

dynamically evolving spatial random fields using exact sampling from matrix-normal and Wishart families. Closed form expressions for the necessary statistical distributions (also using hyper-T distributions) are derived as are expressions for log point-wise predictive or posterior predictive distributions that are used for model selection.

For scaling up emulation, we adapt the FFBS algorithm to Bayesian transfer learning for emulating dynamically evolving mechanistic fields with very large amounts of spatial locations or time points. Here, too, we depart from much of the statistical literature on high-dimensional inference with computer models (Higdon et al., 2008; Gu and Berger, 2016; Gu and Li, 2022) where dimension-reduction is achieved using low-dimensional projections embedded within the statistical models (such as orthogonal latent factor models). Our approach here resembles data partitioning strategies such as treed Gaussian processes (Gramacy and Lee, 2008), meta-kriging (Guhaniyogi and Banerjee, 2018; Guhaniyogi et al., 2017) and predictive stacking over data subsets (Presicce and Banerjee, 2024), but is simpler than in geostatistical settings because the dynamic emulator can be designed executed over regular spatial coordinates and time points to enable sequential updating of the FFBS algorithm over the subsets of the data. For probabilistic learning of the model inputs, we regress the observed field on the emulated field and the mechanistic system parameters are estimated with uncertainty quantification using MCMC but without requiring any additional emulation (using "modularized" inference as described in Bayarri et al., 2007a,b, 2009).

The structure of this article is as follows. Section 2 develops a conjugate family of Bayesian state space models for space-time mechanistic systems emphasizing exact conjugate matrix-variate models to emulate the mechanistic system at arbitrary inputs. Section 3 provides Bayesian model comparison metrics to compare different models for emulating the mechanistic system. Section 4 devises the Bayesian transfer learning approach for scaling up emulation over large spatial fields. Section 6 develops probabilistic learning of mechanistic parameters using field-data and Section 5. Section 7 provide a set of illustrations showing the applications of our dynamic framework using diverse mechanistic systems.

2. Bayesian State-Space Models for Mechanistic Systems

We offer a brief review of Bayesian state-space models (SSMs) – also known as Bayesian dynamic models (DLM) – closely following developments in West and Harrison (1997) and Petris et al. (2009), but adapting or extending familiar distribution theory in these texts to a general multivariate DLM context for our subsequent developments.

2.1 Exact (multivariate) Bayesian inference

We consider mechanistic systems depending on space and time. Examples include broad classes of space-time partial differential equations and, even more generally, all deterministic computer models or agent-based models whose inputs include space and time. We build dynamic emulator models that will allow us to introduce spatial dependence in the output of the mechanistic system. Let $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of N mechanistic system inputs, where each $\boldsymbol{x}_i$ is a d-dimensional vector. Let $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_S\}$ denote the set of observed spatial locations. For any combination $(\boldsymbol{x}, \boldsymbol{s}) \in \mathcal{X} \times \mathcal{S}$, we denote $y_t(\boldsymbol{x}, \boldsymbol{s})$ as the output of the mechanistic system at time point t.

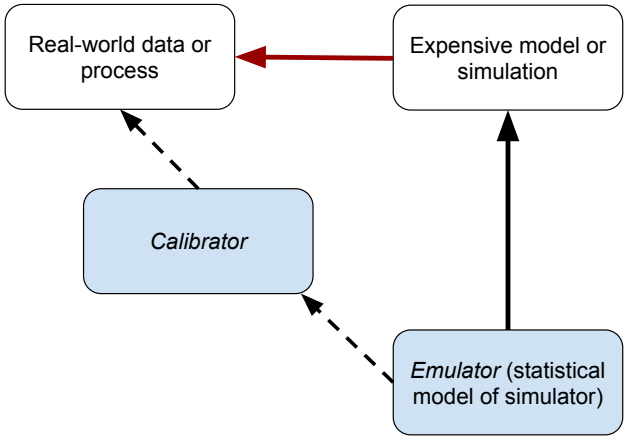

Figure 1: A graphical illustration of the emulation/calibration paradigm. A statistical model emulates the mechanistic system from experimental runs to enable efficient calibration based on observed data. We build the emulator and calibrator modules through a combination of Gaussian and non-Gaussian state-space methods and Gaussian process regression.

A conjugate Matrix-Normal-Inverse-Wishart Bayesian model for space-time settings is

$$\begin{aligned} \boldsymbol{Y}_t &= \boldsymbol{F}_t\boldsymbol{\Theta}_t + \boldsymbol{\mathcal{E}}_t, \quad \boldsymbol{\mathcal{E}}_t \overset{\text{ind.}}{\sim} \mathcal{MN}_{N\times S}(\boldsymbol{O}, \boldsymbol{V}_t, \boldsymbol{\Sigma}), \quad t = 1, 2, \ldots, T \\ \boldsymbol{\Theta}_t &= \boldsymbol{G}_t\boldsymbol{\Theta}_{t-1} + \boldsymbol{\Gamma}_t, \quad \boldsymbol{\Gamma}_t \overset{\text{ind.}}{\sim} \mathcal{MN}_{p\times S}(\boldsymbol{O}, \boldsymbol{W}_t, \boldsymbol{\Sigma}), \quad t = 1, 2, \ldots, T \\ \boldsymbol{\Theta}_0 \mid \boldsymbol{\Sigma} &\sim \mathcal{MN}_{p\times S}(\boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{\Sigma}), \quad \boldsymbol{\Sigma} \sim \mathcal{IW}(n_0, \boldsymbol{D}_0), \end{aligned} \tag{2}$$

where $\boldsymbol{Y}_t$ is $N \times S$ whose (i, j)-th element is $y_t(\boldsymbol{x}_i, \boldsymbol{s}_j)$, $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are $N \times p$ and $p \times p$, respectively, with entries completely known, $\boldsymbol{\Theta}_t$ is $p \times S$ comprising latent effects over space and $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$ are correlation matrices of order $N \times N$ and $p \times p$, respectively, for each $t = 1, 2, \ldots, T$. The important distinction from the more conspicuous univariate DLM setting is that $\boldsymbol{\mathcal{E}}_t$ and $\boldsymbol{\Gamma}_t$ are now matrix-variate normal distributions with zero matrices as their means, $\boldsymbol{V}_t$ and $\boldsymbol{W}_t$ are known correlation matrices modeling dependence across the rows for the respective distributions, and $\boldsymbol{\Sigma}$ is an $S \times S$ covariance matrix modeling the spatial dependence among the columns in $\boldsymbol{\mathcal{E}}_t$ and in $\boldsymbol{\Gamma}_t$. The joint prior density $p(\boldsymbol{\Theta}_0, \boldsymbol{\Sigma})$ in (2) is a Matrix-Normal-Inverse-Wishart, denoted by $\mathcal{MNIW}(\boldsymbol{\Theta}_0, \boldsymbol{\Sigma} \mid \boldsymbol{m}_0, \boldsymbol{M}_0, n_0, \boldsymbol{D}_0)$, with density function

$$\begin{aligned} p(\boldsymbol{\Theta}_0, \boldsymbol{\Sigma}) \propto & \underbrace{\frac{(\det(\boldsymbol{D}_0))^{n_0/2} \exp\left(-\frac{1}{2}\mathrm{tr}\left[\boldsymbol{D}_0\boldsymbol{\Sigma}^{-1}\right]\right)}{2^{n_0 S/2}\Gamma_S(\frac{n_0}{2})\,(\det(\boldsymbol{\Sigma}))^{\frac{n_0+S+1}{2}}}}_{p(\boldsymbol{\Sigma})=\mathcal{IW}(\boldsymbol{\Sigma}|n_0,\boldsymbol{D}_0)} \\ & \times \underbrace{\frac{\exp\left(-\frac{1}{2}\mathrm{tr}\left[(\boldsymbol{\Theta}_0 - \boldsymbol{m}_0)^\top \boldsymbol{M}_0^{-1}(\boldsymbol{\Theta}_0 - \boldsymbol{m}_0)\boldsymbol{\Sigma}^{-1}\right]\right)}{(2\pi)^{pS/2}\,(\det(\boldsymbol{M}_0))^{S/2}\,(\det(\boldsymbol{\Sigma}))^{p/2}}}_{p(\boldsymbol{\Theta}_0|\boldsymbol{\Sigma})=\mathcal{MN}(\boldsymbol{\Theta}_0|\boldsymbol{m}_0,\boldsymbol{M}_0,\boldsymbol{\Sigma})}, \end{aligned} \tag{3}$$

Algorithm 1 Sampling from matrix normal distribution

1: **Input:** $p \times S$ mean matrix $\boldsymbol{m}$, $p \times p$ lower-triangular Cholesky factor $\boldsymbol{L_M}$ of $\boldsymbol{M}$ and $S \times S$ lower-triangular Cholesky factor $\boldsymbol{L_\Sigma}$ of $\boldsymbol{\Sigma}$
2: **Output:** Sample from $\mathcal{MN}_{p\times S}(\boldsymbol{m}, \boldsymbol{M}, \boldsymbol{\Sigma})$
3: **function** SampleMatrixNormal($\boldsymbol{m}, \boldsymbol{L_M}, \boldsymbol{L_\Sigma}$)
4: Draw $\boldsymbol{Z} \sim \mathcal{MN}_{p\times S}(\boldsymbol{O}, \boldsymbol{I}_p, \boldsymbol{I}_S)$ ▷ $\boldsymbol{I}$ is identity matrix, $O(pS)$
5: $\boldsymbol{\Theta} = \boldsymbol{m} + \boldsymbol{L_M Z L_\Sigma^\top}$ ▷ $O(pS^2 + p^2S)$
6: **return** $\boldsymbol{\Theta}$
7: **end function** ▷ $O(pS^2)$

where Γ_S is the S-variate multivariate Gamma function and $\mathrm{tr}(\cdot)$ is the trace function of a matrix. The matrix-normal density of a random matrix corresponds to a multivariate normal density for the vectorized columns of the matrix. For example, the above density $p(\boldsymbol{\Theta}_0 \mid \boldsymbol{\Sigma})$ in (3) is equivalent to $\mathrm{vec}\,(\boldsymbol{\Theta}_0) \mid \boldsymbol{\Sigma} \sim \mathcal{N}_{pS}\,(\mathrm{vec}(\boldsymbol{m}_0), \boldsymbol{\Sigma} \otimes \boldsymbol{M}_0)$, where $\mathrm{vec}(\boldsymbol{\Theta}_0)$ is the $pS \times 1$ vector of the stacked columns (first to last) of $\boldsymbol{\Theta}_0$ and $\otimes$ is the Kronecker product (see, e.g., Banerjee and Roy, 2014, for properties of the vec operator and its connections to the Kronecker product). Standard calculations for multivariate normal distributions yield

$$\begin{aligned} p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:t}) &= \underbrace{\mathcal{IW}(\boldsymbol{\Sigma} \mid n_t, \boldsymbol{D}_t)}_{p(\boldsymbol{\Sigma}\mid \boldsymbol{Y}_{1:t})} \times \underbrace{\mathcal{MN}(\boldsymbol{\Theta}_t \mid \boldsymbol{m}_t, \boldsymbol{M}_t, \boldsymbol{\Sigma})}_{p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma})} \\ &= \mathcal{MNIW}(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t)\ , \end{aligned} \tag{4}$$

where $\boldsymbol{m}_t$, $\boldsymbol{M}_t$, n_t and $\boldsymbol{D}_t$ can be calculated recursively over t as we describe shortly. Sampling $[\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}] \sim \mathcal{MNIW}(\boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t)$ proceeds by first drawing $\boldsymbol{\Sigma} \sim \mathcal{IW}(n_t, \boldsymbol{D}_t)$ and then drawing $\boldsymbol{\Theta}_t \sim \mathcal{MN}(\boldsymbol{m}_t, \boldsymbol{M}_t, \boldsymbol{\Sigma})$ using the drawn value of $\boldsymbol{\Sigma}$.

Algorithm 1 describes the procedure to sample from the matrix-normal distribution using, as input, the mean and the lower-triangular Cholesky factors of the two covariance matrices. We indicate the complexity in terms of the number of floating point operations (flops) for each step in the right-hand column of Algorithm 1. Drawing $\boldsymbol{Z} \sim \mathcal{MN}_{p\times S}(\boldsymbol{O}, \boldsymbol{I}_p, \boldsymbol{I}_S)$ in Line 4 costs $\sim O(pS)$, while Line 5 costs $\sim O(pS^2 + p^2S)$, so the total cost is $O(pS + pS^2 + p^2S) \sim O(pS^2)$ since $p \leq S$ in our subsequent applications.

Closely related to equations (3) and (4) is the Hyper-T distribution. We obtain it by integrating $\boldsymbol{\Sigma}$ from the Matrix-Normal-Inverse-Wishart (Gupta and Nagar, 1999; Zhu et al., 2007). Doing so for, say, equation (4), which we denote by $\mathcal{HT}(\boldsymbol{\Theta}_t | \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t)$, gives us:

$$\begin{aligned} p(\boldsymbol{\Theta}_t | \boldsymbol{Y}_{1:t}) &= \int_{\boldsymbol{\Sigma}>0} \mathcal{MNIW}(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t) d\boldsymbol{\Sigma} \\ &= \frac{\Gamma_S\left(\frac{n_t+p}{2}\right) \left(\det(\boldsymbol{I}_S + \boldsymbol{D}_t^{-1}(\boldsymbol{\Theta} - \boldsymbol{m}_t)^\top \boldsymbol{M}_t^{-1}(\boldsymbol{\Theta} - \boldsymbol{m}_t))\right)^{-\frac{n_t+p}{2}}}{\pi^{pS/2}\Gamma_S\left(\frac{n_t}{2}\right) (\det(\boldsymbol{M}_t))^{S/2} (\det(\boldsymbol{D}_t))^{p/2}} \\ &= \mathcal{HT}(\boldsymbol{\Theta}_t | \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t) \end{aligned} \tag{5}$$

Algorithm 2 Sampling from hyper-T distribution

1: **Input:** $p \times S$ matrix $\boldsymbol{m}$, row-covariance matrix $p \times p$ matrix $\boldsymbol{M}$, positive integer degrees of freedom n, positive symmetric definite $S \times S$ scale matrix $\boldsymbol{D}$
2: **Output:** Sample from $\mathcal{HT}_{p\times S}(\boldsymbol{m}, \boldsymbol{M}, n, \boldsymbol{D})$
3: **function** SampleHyperT($\boldsymbol{m}, \boldsymbol{M}, n, \boldsymbol{D}$)
4: Draw $\boldsymbol{\Sigma} \sim \mathcal{IW}_{S\times S}(n, \boldsymbol{D})$ ▷ $O(S^3)$
5: $\boldsymbol{L}_{\boldsymbol{M}} \leftarrow$ Cholesky($\boldsymbol{M}$) ▷ $O(p^3)$
6: $\boldsymbol{L}_{\boldsymbol{\Sigma}} \leftarrow$ Cholesky($\boldsymbol{\Sigma}$) ▷ $O(S^3)$
7: $\boldsymbol{\Theta} \leftarrow$ SampleMatrixNormal($\boldsymbol{m}, \boldsymbol{L}_{\boldsymbol{M}}, \boldsymbol{L}_{\boldsymbol{\Sigma}}$) ▷ Algorithm 1
8: **return** $\boldsymbol{\Theta}$
9: **end function** ▷ $O(S^3)$

The canonical form given in (Gupta and Nagar, 1999) or (Zhu et al., 2007) can be obtained by setting the degrees of freedom for the inverse-Wishart $n_t \leftarrow n_t + S - 1$, implying that the hyper-T is the matrix-T distribution with degrees of freedom $n_t - S + 1$. (It is thus important to point out that the degrees of freedom of the inverse-Wishart n_t is not automatically equal to the degrees of freedom for the canonical form of the matrix-T.) Each entry of the matrix-T random variable is a univariate t-distribution with degrees of freedom $n_t - S + 1$, mean $m_{t,ij}$, and scale term $D_{t,ii}M_{t,jj}/n_t$.

Algorithm 2 describes the procedure to sample from the $\mathcal{HT}_{p\times S}(\boldsymbol{m}, \boldsymbol{M}, n, \boldsymbol{D})$. As in Algorithm 1, the right-hand side column provide the complexity in flops. Lines-4, 5, and 6 together cost $\sim O(p^3 + 2S^3)$ and Line-7 costs $\sim O(pS^2)$, which brings the total cost to $\sim O(p^3 + 2S^3 + pS^2) \sim O(S^3)$ since $p \leq S$ in our subsequent applications.

Also important to consider is a matrix-normal $\boldsymbol{\Theta}_t$, but where $\Sigma = \sigma^2\boldsymbol{R}$, where σ^2 is distributed as inverse-gamma ($\mathcal{IG}(n_t, d_t)$) and $\boldsymbol{R}$ being a given matrix. We denote this density by $\mathcal{MNIG}(\boldsymbol{\Theta}_t, \sigma^2 \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, d_t, \boldsymbol{R})$. Next, we marginalize out the σ^2 to obtain a multivariate t-distribution, but one in which the mean arguments are matrices rather than vectors. We call this the hyper-T-scalar and denote it by $\mathcal{HT}_s(\boldsymbol{\Theta}_t \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, d_t, \boldsymbol{R})$:

$$
\begin{aligned}
p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:t}, \boldsymbol{R}) =& \int_{\sigma^2>0} \mathcal{MNIG}(\boldsymbol{\Theta}_t, \sigma^2 \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, d_t, \boldsymbol{R}) d\sigma^2 \\
=& \frac{\Gamma(\frac{pS}{2} + n_t)}{\Gamma(n_t)} (2\pi d_t)^{-\frac{pS}{2}} |\boldsymbol{M}_t|^{-\frac{S}{2}} |\boldsymbol{R}|^{-\frac{p}{2}} \times \\
& \left| 1 + \frac{1}{2d_t} \mathrm{tr}\left[\boldsymbol{R}^{-1} (\boldsymbol{\Theta}_t - \boldsymbol{m}_t)^\top \boldsymbol{M}_t^{-1} (\boldsymbol{\Theta}_t - \boldsymbol{m}_t) \right] \right|^{-\left(\frac{pS}{2} + n_t\right)} \\
=& \mathcal{HT}_s(\boldsymbol{\Theta}_t \mid \boldsymbol{m}_t, \boldsymbol{M}_t, n_t, d_t, \boldsymbol{R})
\end{aligned}
\tag{6}
$$

The matrix-variate SSM in (2) delivers posterior and predictive distributions in closed form. Exact Bayesian inference is possible by sampling from the posterior distribution $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$ using the Forward Filter Backward Sampling algorithm (Carter and Kohn, 1994; Frühwirth-Schnatter, 2006). The basic idea is fairly simple. We first move forward in time up to $t = T$ and draw one instance of $(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_T)$ from $p(\boldsymbol{\Theta}_T, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$. This is the "forward filtering" step ("FF"). Next, we execute "backward sampling" (BS): for each $t = T-1, T-2, \ldots, 0$, we draw one instance of $\boldsymbol{\Theta}_t$ from $p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t})$. This entire

Algorithm 3 Kalman (forward) filter

1: **Input:** Data $\boldsymbol{Y}_{1:T}$, hyperparameters n_0, $\boldsymbol{D}_0$, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$, correlation matrices $\boldsymbol{V}_{1:T}$, $\boldsymbol{W}_{1:T}$,
2: observation and state transition matrices $\boldsymbol{F}_{1:T}$ and $\boldsymbol{G}_{1:T}$.
3: **Output:** Filtering distribution parameters at time $t = 0, \ldots, T$
4: **function** FILTER($\boldsymbol{Y}_{1:T}, n_0, \boldsymbol{D}_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{G}_{1:T}, \boldsymbol{F}_{1:T}, \boldsymbol{V}_{1:T}, \boldsymbol{W}_{1:T}$)
5: **for** $t = 1$ to T **do**
6: *# Compute prior distribution* $[\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}_t \mid \boldsymbol{Y}_{1:t-1}] \sim \mathcal{MNIW}(\boldsymbol{a}_t, \boldsymbol{A}_t, n_t^*, \boldsymbol{D}_t^*)$:
7: $\boldsymbol{a}_t \leftarrow \boldsymbol{G}_t \boldsymbol{m}_{t-1}, \boldsymbol{A}_t \leftarrow \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{W}_t, n_t^* \leftarrow n_{t-1}, \boldsymbol{D}_t^* \leftarrow \boldsymbol{D}_{t-1}$ ▷ $O(p^3 + p^2 S)$
8: *# Compute one-step-ahead forecast* $p(\boldsymbol{Y}_t | \boldsymbol{Y}_{1:t-1}) \sim \mathcal{HT}(\boldsymbol{F}_t \boldsymbol{a}_t, \boldsymbol{F}_t \boldsymbol{A}_t \boldsymbol{F}_t + \boldsymbol{V}_t, n_t^*, \boldsymbol{D}_t^*)$
9: $\boldsymbol{q}_t \leftarrow \boldsymbol{F}_t \boldsymbol{a}_t, \boldsymbol{Q}_t \leftarrow \boldsymbol{F}_t \boldsymbol{A}_t \boldsymbol{F}_t^\top + \boldsymbol{V}_t$ ▷ $O(pSN + p^2 N + N^2)$
10: *# Compute filtering distribution* $p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}_t \mid \boldsymbol{Y}_{1:t}) \sim \mathcal{MNIW}(\boldsymbol{m}_t, \boldsymbol{M}_t, n_t, \boldsymbol{D}_t)$:
11: $\boldsymbol{m}_t \leftarrow \boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{Y}_t - \boldsymbol{q}_t),\ \boldsymbol{M}_t \leftarrow \boldsymbol{A}_t - \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{F}_t \boldsymbol{A}_t^\top$
12: ▷ $O(pN^2 + p^2 N + p^3 + pSN + N^3)$
13: $n_t \leftarrow n_t^* + N,\ \boldsymbol{D}_t \leftarrow \boldsymbol{D}_t^* + (\boldsymbol{Y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{Y}_t - \boldsymbol{q}_t)$ ▷ $O(SN^2 + S^2 N)$
14: **end for**
15: **return** $\{n_t, \boldsymbol{D}_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t\}_{t=0}^T$
16: **end function** ▷ $O(T(p^2 S + S^2 N))$

process yields one instance of $(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T})$ from $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$. Backward sampling makes use of the Markovian structure in (2), which implies that

$$p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T}) = p(\boldsymbol{\Theta}_T, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T}) \prod_{t=0}^{T-1} p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t}) \,. \tag{7}$$

The "FF" followed by the "BS" steps complete one iteration of the FFBS algorithm yielding one draw from $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$. In essence, the FFBS algorithm first applies the Kalman filter of Algorithm 3 before initializing the backward sampling through the Kalman or Rauch–Tung–Striebel smoother (Rauch et al., 1965; Särkkä and Svensson, 2013). Repeating the FFBS cycle L times yields L posterior samples of $\boldsymbol{\Theta}_{0:T}$ and $\boldsymbol{\Sigma}$. In object-oriented computing paradigms, we can execute L draws of $(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_T)$ at the end of FF. Then, for each drawn $(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_T)$ we execute $T - 1$ draws of the BS to obtain L posterior samples.

The specific FFBS algorithm to estimate (2) accomplishes the forward filtering step by recursively computing the following quantities for each $t = 1, \ldots, T$,

$$\begin{aligned} &\boldsymbol{a}_t = \boldsymbol{G}_t \boldsymbol{m}_{t-1}; \quad \boldsymbol{A}_t = \boldsymbol{G}_t \boldsymbol{M}_{t-1} \boldsymbol{G}_t^\top + \boldsymbol{W}_t; \quad \boldsymbol{q}_t = \boldsymbol{F}_t \boldsymbol{a}_t; \quad \boldsymbol{Q}_t = \boldsymbol{F}_t \boldsymbol{A}_t \boldsymbol{F}_t^\top + \boldsymbol{V}_t; \\ &\boldsymbol{m}_t = \boldsymbol{a}_t + \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{Y}_t - \boldsymbol{q}_t); \quad \boldsymbol{M}_t = \boldsymbol{A}_t - \boldsymbol{A}_t \boldsymbol{F}_t^\top \boldsymbol{Q}_t^{-1} \boldsymbol{F}_t \boldsymbol{A}_t; \\ &n_t = n_{t-1} + N; \quad \text{and} \quad \boldsymbol{D}_t = \boldsymbol{D}_{t-1} + (\boldsymbol{Y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{Y}_t - \boldsymbol{q}_t) \end{aligned} \tag{8}$$

Algorithm 3 outlines the steps for forward filtering. The right-hand column provides the computational complexity in terms of flops. Lines-7, 9, 11, and 13 together complete one iteration of the for-loop, with a combined cost of $\sim O(p^3 + p^2(S+N) + p(SN + N^2) + S^2 N + SN^2 + N^3)$. In particular, if $p, N \leq S$, then the combined cost is $\sim O(p^2 S + S^2 N)$ for each t. This yields a total cost of $\sim O(T(p^2 S + S^2 N))$. We provide the costs in our subsequent algorithms using $p, N \leq S$. This is true for all our applications except the predator-prey system in Section 5.1.

Algorithm 4 Backward sampler

Input: Filtering parameters and inputs from Algorithm 3
Output: L posterior samples from $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$
function BACKWARDSAMPLE($\{n_t, \boldsymbol{D}_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t, \boldsymbol{G}_t\}_{t=0}^{T}$, L)
  Draw L samples from $\boldsymbol{\Sigma} \sim \mathcal{IW}(n_T, \boldsymbol{D}_T)$ ▷ $O(LS^3)$
  Draw L samples from $\boldsymbol{\Theta}_T \sim \mathcal{MN}(\boldsymbol{m}_T, \boldsymbol{M}_T, \boldsymbol{\Sigma})$ ▷ $O(LS^3)$
  $\boldsymbol{h}_T \leftarrow \boldsymbol{m}_T, \boldsymbol{H}_T \leftarrow \boldsymbol{M}_T$
  for $t = T-1$ to 1 **do**
    $\boldsymbol{h}_t \leftarrow \boldsymbol{m}_t + \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{h}_{t+1} - \boldsymbol{a}_{t+1})$ ▷ $O(p^3 + p^2 S)$
    $\boldsymbol{H}_t \leftarrow \boldsymbol{M}_t - \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{A}_{t+1} - \boldsymbol{H}_{t+1}) \boldsymbol{A}_{t+1}^{-1} \boldsymbol{G}_{t+1} \boldsymbol{M}_t$ ▷ $O(p^3)$
    Draw L samples from $\boldsymbol{\Theta}_t \sim \mathcal{MN}(\boldsymbol{h}_t, \boldsymbol{H}_t, \boldsymbol{\Sigma})$ ▷ $O(LS^3)$
  end for
  return $\{\boldsymbol{h}_{1:T}, \boldsymbol{H}_{1:T}\}$ and L samples of $\{\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma}\}$
end function ▷ $O(TLS^3)$

We then draw one instance of $(\boldsymbol{\Theta}_T, \boldsymbol{\Sigma}) \mid \boldsymbol{Y}_{1:T} \sim \mathcal{MNIW}(\boldsymbol{m}_T, \boldsymbol{M}_T, n_T, \boldsymbol{D}_T\cdot)$, where $\boldsymbol{a}_t$ and $\boldsymbol{m}_t$ are both $p \times S$ and $\boldsymbol{q}_t$ is $N \times S$. For the inverse-Wishart parameters, $n_t = n_{t-1} + N$ is a scalar and $\boldsymbol{D}_t = \boldsymbol{D}_{t-1} + (\boldsymbol{Y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1} (\boldsymbol{Y}_t - \boldsymbol{q}_t)$ is $S \times S$. An exact FFBS algorithm proceeds by forward filtering to $t = T$ to draw L samples from $p(\boldsymbol{\Theta}_T, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$ from (4). Next, for backward sampling we exploit standard Bayesian probability calculations to note that $(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}) \mid \boldsymbol{Y}_{1:T}) \sim \mathcal{MNIW}(\boldsymbol{h}_t, \boldsymbol{H}_t, n_T, \boldsymbol{D}_T)$, where $\boldsymbol{h}_T = \boldsymbol{m}_T$, $\boldsymbol{H}_T = \boldsymbol{M}_T$, and

$$\begin{aligned} \boldsymbol{h}_t &= \boldsymbol{m}_t + \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{h}_{t+1} - \boldsymbol{a}_{t+1}) \quad \text{and} \\ \boldsymbol{H}_t &= \boldsymbol{M}_t - \boldsymbol{M}_t \boldsymbol{G}_{t+1}^\top \boldsymbol{A}_{t+1}^{-1} (\boldsymbol{A}_{t+1} - \boldsymbol{H}_{t+1}) \boldsymbol{A}_{t+1}^{-1} \boldsymbol{G}_{t+1} \boldsymbol{M}_t \, . \end{aligned} \tag{9}$$

Therefore, for each $t = T-1, \ldots, 0$ we draw one value of $\boldsymbol{\Theta}_t \sim \mathcal{MN}(\boldsymbol{h}_t, \boldsymbol{H}_t, \boldsymbol{\Sigma})$ for each of the L values of $\boldsymbol{\Sigma}$ at the end of the "FF" step, where $\boldsymbol{h}_t$ and $\boldsymbol{H}_t$ are computed using (9). The end of the FFBS yields L posterior samples from $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$.

Algorithm 5 Forward-filter-backward-sampler algorithm

Input: Data $\boldsymbol{Y}_{1:T}$ and Kalman filter starting values n_0, $\boldsymbol{D}_0$, $\boldsymbol{m}_0$, $\boldsymbol{M}_0$
  Observation and state transition matrices $\boldsymbol{F}_{1:T}$ and $\boldsymbol{G}_{1:T}$
  Correlation matrices $\boldsymbol{V}_{1:T}$ and $\boldsymbol{W}_{1:T}$, number of samples L
Output: Sample from posterior $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} | \boldsymbol{Y}_{1:T})$
function FFBS($\boldsymbol{Y}_{1:T}, n_0, \boldsymbol{D}_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{F}_{1:T}, \boldsymbol{G}_{1:T}, \boldsymbol{V}_{1:T}, \boldsymbol{W}_{1:T}, L$)
  $\texttt{FF}_{\texttt{Out}} \leftarrow \texttt{Filter}(\boldsymbol{Y}_{1:T}, n_0, \boldsymbol{D}_0, \boldsymbol{m}_0, \boldsymbol{M}_0, \boldsymbol{G}_{1:T}, \boldsymbol{F}_{1:T}, \boldsymbol{V}_{1:T}, \boldsymbol{W}_{1:T})$ ▷ Algorithm 3
  $\texttt{BS}_{\texttt{Out}} \leftarrow \texttt{BackwardSample}(\{n_t, \boldsymbol{D}_t, \boldsymbol{a}_t, \boldsymbol{A}_t, \boldsymbol{m}_t, \boldsymbol{M}_t, \boldsymbol{G}_t\}_{t=0}^{T}, L)$ ▷ Algorithm 4
  return $\texttt{BS}_{\texttt{Out}} = \{\boldsymbol{h}_{1:T}, \boldsymbol{H}_{1:T}, \{\boldsymbol{\Theta}_{0:T}^{(l)}, \boldsymbol{\Sigma}^{(l)}\}_{l=1}^{L}\}$
end function ▷ $O(TLS^3)$

Algorithm 4 outlines the steps for backward sampling, with the computational complexity detailed in the right-hand column in terms of flops. Lines-4 and 5 each draw L samples of $\boldsymbol{\Sigma}$ and $\boldsymbol{\Theta}_T$, respectively, with a computational cost of $\sim O(LS^3)$ for both. Meanwhile, Lines-8, 9, and 10 together constitute one iteration of the for-loop, with a cost of $\sim$

$O(p^3 + p^2S + LS^3) \sim O(LS^3)$. This results in a total cost of $\sim O(TLS^3)$ assuming $p, N \leq S$. Algorithm 5 presents the complete forward-filter-backward-sampler algorithm, incorporating both Algorithm 3 and Algorithm 4. The overall computational cost is $\sim O(T(p^2S + S^2N + LS^3)) \sim O(TLS^3)$.

Bayesian predictive inference is also convenient if we seek to predict or impute the values in a new $\tilde{N} \times S$ matrix $\tilde{\boldsymbol{Y}}_t$ corresponding to a new set of mechanistic inputs $\tilde{\mathcal{X}}$, given the $\tilde{N} \times p$ matrix $\tilde{\boldsymbol{F}}_t$ for any given t, where $\tilde{N}$ represents the number of new mechanistic system inputs. The predictive distribution for $\tilde{\boldsymbol{Y}}_t$ follows from the augmented model,

$$\begin{bmatrix} \boldsymbol{Y}_t \\ \tilde{\boldsymbol{Y}}_t \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_t \\ \tilde{\boldsymbol{F}}_t \end{bmatrix} \boldsymbol{\Theta_t} + \begin{bmatrix} \boldsymbol{\mathcal{E}_t} \\ \tilde{\boldsymbol{\mathcal{E}}}_t \end{bmatrix}, \quad \begin{bmatrix} \boldsymbol{\mathcal{E}}_t \\ \tilde{\boldsymbol{\mathcal{E}}}_t \end{bmatrix} \overset{\text{ind.}}{\sim} \mathcal{MN}_{(N+\tilde{N})\times S} \left(\begin{bmatrix} \boldsymbol{O}_{N\times S} \\ \boldsymbol{O}_{\tilde{N}\times S} \end{bmatrix}, \begin{bmatrix} \boldsymbol{V}_t & \boldsymbol{J}_t \\ \boldsymbol{J}_t{}^\top & \tilde{\boldsymbol{V}}_t \end{bmatrix}, \boldsymbol{\Sigma} \right), \tag{10}$$

for $t = 1, 2, \ldots, T$. Therefore, for each drawn value of $\{\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}\} \sim p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$ from the FFBS algorithm, we draw one instance of $\tilde{\boldsymbol{Y}}_t$ from the conditional predictive density

$$p(\tilde{\boldsymbol{Y}}_t \mid \boldsymbol{Y}_{1:T}, \boldsymbol{\Theta_t}, \boldsymbol{\Sigma}) = \mathcal{MN}(\tilde{\boldsymbol{Y}}_t \mid \tilde{\boldsymbol{F}}_t\boldsymbol{\Theta}_t + \boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}_t\boldsymbol{\Theta}_t), \tilde{\boldsymbol{V}}_t - \boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1}\boldsymbol{J}_t, \boldsymbol{\Sigma}) \,. \tag{11}$$

Repeating this for all the posterior samples of $\{\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma}\}$ will yield samples of $\tilde{\boldsymbol{Y}}_{1:T}$ from the posterior predictive distribution $p(\tilde{\boldsymbol{Y}}_{1:T} \mid \boldsymbol{Y}_{1:T})$. Therefore, predictive inference can be carried out using (11) for arbitrary inputs in $\tilde{\mathcal{X}}$ using stored posterior samples from the training data. The full expression for the posterior predictive density is itself a Hyper-T, with the following expression:

$$\begin{aligned} p(\tilde{\boldsymbol{Y}}_t \mid \boldsymbol{Y}_{1:T}) =& \mathcal{HT}(\tilde{\boldsymbol{Y}}_t \mid \tilde{\boldsymbol{F}}_t\boldsymbol{h}_t + \boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}_t\boldsymbol{h}_t), \tilde{\boldsymbol{F}}_t\boldsymbol{H}_t\tilde{\boldsymbol{F}}_t^\top + \tilde{\boldsymbol{V}}_t - \\ & (\boldsymbol{F}_t\boldsymbol{H}_t\tilde{\boldsymbol{F}}_t^\top + \boldsymbol{J}_t)^\top (\boldsymbol{F}_t\boldsymbol{H}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t)^{-1}(\boldsymbol{F}_t\boldsymbol{H}_t\tilde{\boldsymbol{F}}_t + \boldsymbol{J}_t), n_T, \boldsymbol{D}_T) \end{aligned} \tag{12}$$

The same conclusions we draw as for equation (5) lead us to conclude that each entry of $\tilde{\boldsymbol{Y}}_t$, $\tilde{\boldsymbol{Y}}_{t,ij}$, is a univariate-t, with mean $\tilde{\boldsymbol{F}}_{t,i:}\boldsymbol{h}_{t,:j} + \boldsymbol{J}_{t,:i}^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_{t,:j} - \boldsymbol{F}_t\boldsymbol{h}_{t,:j})$ and scale parameter $(\tilde{\boldsymbol{F}}_{t,j:}\boldsymbol{H}_t\tilde{\boldsymbol{F}}_{t,j:}^\top + \tilde{\boldsymbol{V}}_{t,jj} - (\boldsymbol{F}_t\boldsymbol{H}_t\tilde{\boldsymbol{F}}_{t,:j}^\top + \boldsymbol{J}_{t,:j})^\top (\boldsymbol{F}_t\boldsymbol{H}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t)(\boldsymbol{F}_t\boldsymbol{H}_t\tilde{\boldsymbol{F}}_{t,j:}^\top + \boldsymbol{J}_{t,:j}))(\boldsymbol{D}_{T,ii})/n_T$.

We do not require fresh training of the model for predictive inference provided the distribution in (10) is valid—a matter we turn to next.

2.2 Mechanistic emulation using Gaussian Processes

Emulating the mechanistic system requires that we train the model in (2) from the inputs in $\mathcal{X}$ using the FFBS algorithm and, subsequently, to predict or interpolate the outcome matrix for possibly new inputs in $\tilde{\mathcal{X}}$. For predictive inference, we must ensure that the distribution in (10) is well-defined, which, in turn, requires that $\begin{bmatrix} \boldsymbol{V}_t & \boldsymbol{J}_t \\ \boldsymbol{J}_t{}^\top & \tilde{\boldsymbol{V}}_t \end{bmatrix}$ is positive definite for arbitrary mechanistic inputs $\mathcal{X} \cup \tilde{\mathcal{X}}$. This will be ensured if we ensure that each column of the augmented outcome matrix $\begin{bmatrix} \boldsymbol{Y}_t \\ \tilde{\boldsymbol{Y}}_t \end{bmatrix}$ is a realization of a valid stochastic process over $\mathcal{X} \cup \tilde{\mathcal{X}}$. A customary choice for mechanistic emulation is the Gaussian process.

The Gaussian process acts as a prior on of $\begin{bmatrix} \boldsymbol{Y}_t \\ \tilde{\boldsymbol{Y}}_t \end{bmatrix}$. Thus, for each of the S locations (or column indices), we assume $y_t(\boldsymbol{x}, \boldsymbol{s}) \overset{ind}{\sim} \mathcal{GP}\left(\mu_t(\boldsymbol{x}, \boldsymbol{s}), C(\cdot, \cdot; \boldsymbol{\beta})\right)$, where the mean function

$\mu_t(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{f}_t(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{\Theta}_t(\boldsymbol{s})$ and a positive definite correlation function $C(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\beta})$. This yields, $\boldsymbol{V}_t = (C(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{\beta}))$ for pairs of inputs in $\mathcal{X}$, $\tilde{\boldsymbol{V}}_t = (C(\tilde{\boldsymbol{x}}_i, \tilde{\boldsymbol{x}}_j; \boldsymbol{\beta}))$ for pairs of inputs in $\tilde{\mathcal{X}}$ and $\boldsymbol{J}_t = (C(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_j; \boldsymbol{\beta}))$ for $\boldsymbol{x}_i \in \mathcal{X}$ and $\tilde{\boldsymbol{x}}_j$ in $\tilde{\mathcal{X}}$. This specification ensures that (10) is a valid probability distribution by virtue of $C(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\beta})$ being a valid correlation function. Several options for valid correlation functions are available, including the rich Matérn class used widely in spatial statistics. In mechanistic emulation, we are mostly concerned with interpolation so any valid correlation function of a simpler form will suffice. For our current illustrations, we use the squared exponential function

$$C(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\beta}) = \exp\left(-\sum_{i=1}^{d} \beta_i (x_i - x_i')^2\right), \tag{13}$$

where the range parameter β_i controls the decay of correlation along the i-th dimension of $\boldsymbol{\beta}$. In particular, let us denote $\boldsymbol{Y}_t$ and $\tilde{\boldsymbol{Y}}_t$ as $\boldsymbol{Y}_t(\mathcal{X})$ and $\tilde{\boldsymbol{Y}}_t(\tilde{\mathcal{X}})$, respectively. Then, the predictive mean $\mathbb{E}[\boldsymbol{Y}_t(\tilde{\mathcal{X}}) \mid \boldsymbol{Y}_t(\mathcal{X}), \boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma}]$ in (11) acts as an interpolator over the mechanistic inputs. More precisely, if $\tilde{\mathcal{X}} \subseteq \mathcal{X}$, then the rows of $\boldsymbol{J}_t^\top$ are a subset of the rows of $\boldsymbol{V}_t$, which implies that $\boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1} = \boldsymbol{I}(\tilde{\mathcal{X}})$ (since $\boldsymbol{V}_t \boldsymbol{V}_t^{-1} = \boldsymbol{I}(\mathcal{X})$), where $\boldsymbol{I}(\tilde{\mathcal{X}})$ is the subset of rows of the identity matrix indexed by $\tilde{\mathcal{X}}$. Therefore, $\boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}_t \boldsymbol{\Theta}_t) = \boldsymbol{Y}_t(\tilde{\mathcal{X}}) - \tilde{\boldsymbol{F}}_t \boldsymbol{\Theta}_t$ and

$$\mathbb{E}[\boldsymbol{Y}_t(\tilde{\mathcal{X}}) \mid \boldsymbol{Y}_t(\mathcal{X}), \boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma}] = \tilde{\boldsymbol{F}}_t \boldsymbol{\Theta}_t + \boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}_t \boldsymbol{\Theta}_t) = \tilde{\boldsymbol{F}}_t \boldsymbol{\Theta}_t + \boldsymbol{Y}_t(\tilde{\mathcal{X}}) - \tilde{\boldsymbol{F}}_t \boldsymbol{\Theta}_t = \boldsymbol{Y}_t(\tilde{\mathcal{X}}) \, .$$

Furthermore, this interpolation is deterministic because

$$\mathrm{Var}[\boldsymbol{Y}_t(\tilde{\mathcal{X}}) \mid \boldsymbol{Y}_t(\mathcal{X}), \boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma}] = \tilde{\boldsymbol{V}}_t - \boldsymbol{J}_t^\top \boldsymbol{V}_t^{-1} \boldsymbol{J}_t = \tilde{\boldsymbol{V}}_t - \boldsymbol{I}(\tilde{\mathcal{X}}) \boldsymbol{J}_t = \tilde{\boldsymbol{V}}_t - \tilde{\boldsymbol{V}}_t = \boldsymbol{O} \, ,$$

where the second to last equation follows from the fact that $\boldsymbol{I}(\tilde{\mathcal{X}}) \boldsymbol{J}_t = \tilde{\boldsymbol{V}}_t$ when $\tilde{\mathcal{X}} \subseteq \mathcal{X}$.

2.3 Limitations

If we are able to evaluate and store the output from a space-time mechanistic system over a fixed set of S locations and T time points, then we are able to learn about the temporal evolution of the process and the association over the spatial locations (through $\boldsymbol{\Sigma}$) using a conjugate Matrix-variate dynamic linear model framework. A clear advantage of this approach is that inference from the FFBS samples from the exact posterior and posterior predictive distributions and there is no need to resort to more computationally expensive iterative algorithms (e.g., MCMC, INLA, VB) that will require diagnosis of convergence.

The relative simplicity of this learning framework, however, comes with some severe limitations with respect to spatial modeling. First, and perhaps most obviously, the above approach uses an unstructured $\boldsymbol{\Sigma}$ to model the associations across space, whereas it could be argued that one should model $\boldsymbol{\Sigma}$ further using conventional geostatistical kernels that more explicitly model association as a function of the locations (or distances between them). Second, since the dimension of $\boldsymbol{\Sigma}$ is fixed, learning of the mechanistic system occurs only over the fixed set of S spatial locations that have been determined at the design stage to run the system but precludes predictions at arbitrary new locations. Third, it does not accommodate the possibility that for each spatial location, we may not have the same inputs (breaking the $N \times S$ matrix structure) which is the likely scenario when our learning framework includes information from field data that we use to calibrate the mechanistic system.

A more flexible approach builds a learning system that models both the mechanistic system and the spatial associations using stochastic processes with Gaussian processes being the customary choice here. Now $\boldsymbol{\Sigma}$ is constructed using a spatial covariance kernel which leads to a sparser parametrization, easier interpretation, and accommodates predictions over space. While the exact distribution theory for the $S \times S$ covariance matrix $\boldsymbol{\Sigma}$ is lost, we arrive at a richer and more flexible framework that also eliminates the requirement for the dimension of $\boldsymbol{Y}_t$ to be the same for each t. Therefore, we construct

$$\begin{aligned} \boldsymbol{Y}_t &= \boldsymbol{F}_t\boldsymbol{\Theta}_t + \boldsymbol{\mathcal{E}}_t, \quad \boldsymbol{\mathcal{E}}_t \overset{\text{ind.}}{\sim} \sim \mathcal{MN}_{N_t\times S}(\boldsymbol{O}, \boldsymbol{V}_t, \sigma^2\boldsymbol{R}), \\ \boldsymbol{\Theta}_t &= \boldsymbol{G}_t\boldsymbol{\Theta}_{t-1} + \boldsymbol{\Gamma}_t, \quad \boldsymbol{\Gamma}_t \overset{\text{ind.}}{\sim} \mathcal{MN}_{p_t\times S}(\boldsymbol{O}, \boldsymbol{W}_t, \sigma^2\boldsymbol{R}), \\ \boldsymbol{\Theta}_0 \mid \sigma^2 &\sim \mathcal{MN}_{p_t\times S}(\boldsymbol{m}_0, \boldsymbol{M}_0, \sigma^2\boldsymbol{R}), \quad \sigma^2 \sim \mathcal{IG}(n_0, d_0), \end{aligned} \tag{14}$$

where $\boldsymbol{Y}_t$ and $\boldsymbol{\Theta}_t$ are $N_t \times S$ and $p_t \times S$, respectively, $\boldsymbol{F}_t$ and $\boldsymbol{G}_t$ are $N_t \times p_t$ and $p_t \times p_{t-1}$, respectively, $\boldsymbol{R}$ is $S \times S$ spatial correlation matrices whose (i, j)-th elements are values of some spatial correlation function $\rho_t(\boldsymbol{s}_i, \boldsymbol{s}_j)$ with fixed process parameters. Fixing $\boldsymbol{V}_t$, $\boldsymbol{W}_t$, and $\boldsymbol{R}$ in (14) will, again, yield conjugate Bayesian posterior distributions obtained using an FFBS algorithm.

The FFBS algorithm to estimate (14) adapts Algorithms 3, 4 and 5 by recursively computing the following quantities for each $t = 1, \ldots, T$,

$$\begin{aligned} &\boldsymbol{a}_t = \boldsymbol{G}_t\boldsymbol{m}_{t-1}; \quad \boldsymbol{A}_t = \boldsymbol{G}_t\boldsymbol{M}_{t-1}\boldsymbol{G}_t^\top + \boldsymbol{W}_t; \quad \boldsymbol{q}_t = \boldsymbol{F}_t\boldsymbol{a}_t; \quad \boldsymbol{Q}_t = \boldsymbol{F}_t\boldsymbol{A}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t; \\ &\boldsymbol{m}_t = \boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\left(\boldsymbol{Y}_t - \boldsymbol{q}_t\right); \quad \boldsymbol{M}_t = \boldsymbol{A}_t - \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\boldsymbol{F}_t\boldsymbol{A}_t; \\ &n_t = n_{t-1} + \frac{NS}{2}; \quad \text{and} \quad d_t = d_{t-1} + \frac{1}{2}\text{tr}\left[\left(\boldsymbol{Y}_t - \boldsymbol{q}_t\right)^\top\boldsymbol{Q}_t^{-1}\left(\boldsymbol{Y}_t - \boldsymbol{q}_t\right)\boldsymbol{R}^{-1}\right]. \end{aligned} \tag{15}$$

For the inverse-Gamma parameters, n_t is the shape and d_t is the rate. As earlier, we use forward filtering up to $t = T$ to draw L samples from $p(\boldsymbol{\Theta}_T, \sigma^2 \mid \boldsymbol{Y}_{1:T})$ from $\mathcal{MNIG}(\boldsymbol{m}_T, \boldsymbol{M}_T, n_T, d_T)$, where $\boldsymbol{a}_t$ and $\boldsymbol{m}_t$ are both $p_t \times S$ and $\boldsymbol{q}_t$ is $N_t \times S$. For backward sampling, we note that $(\boldsymbol{\Theta}_t, \sigma^2) \mid \boldsymbol{Y}_{1:T}) \sim \mathcal{MNIG}(\boldsymbol{h}_t, \boldsymbol{H}_t, n_T, d_T)$, where $\boldsymbol{h}_T = \boldsymbol{m}_T$, $\boldsymbol{H}_T = \boldsymbol{M}_T$, and

$$\begin{aligned} \boldsymbol{h}_t &= \boldsymbol{m}_t + \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}(\boldsymbol{h}_{t+1} - \boldsymbol{a}_{t+1}) \quad \text{and} \\ \boldsymbol{H}_t &= \boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\left(\boldsymbol{A}_{t+1} - \boldsymbol{H}_{t+1}\right)\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t\,. \end{aligned} \tag{16}$$

Therefore, for each $t = T-1, \ldots, 0$ we draw one value of $\boldsymbol{\Theta}_t \sim \mathcal{MN}(\boldsymbol{h}_t, \boldsymbol{H}_t, \sigma^2\boldsymbol{R})$ for each of the L values of σ^2 at the end of the "FF" step, where $\boldsymbol{h}_t$ and $\boldsymbol{H}_t$ are computed using (16). The end of the FFBS yields L posterior samples from $p(\boldsymbol{\Theta}_{0:T}, \sigma^2 \mid \boldsymbol{Y}_{1:T})$.

3. Bayesian Model Comparisons

3.1 Widely applicable information criteria

We evaluate predictive accuracy of our models using the Watanabe-Akaike (Widely Applicable) Information Criteria (WAIC) defined as

$$\text{WAIC} = -2(\text{lppd} - p_{\text{WAIC}}) \tag{17}$$

where $\text{lppd} = \sum_{t=1}^{T} \log \mathbb{E}_{\boldsymbol{\Theta}_t|\boldsymbol{Y}_{1:T}}\left[p(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t)\right]$ and $p_{\text{WAIC}} = \sum_{t=1}^{T} \text{Var}_{\boldsymbol{\Theta}_t|\boldsymbol{Y}_{1:T}}\left[\log p(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t)\right]$, where we use $\boldsymbol{Y}_t^*$ for the posterior predictive density $p(\boldsymbol{Y}_t^* \mid \boldsymbol{Y}_{1:T})$. The "lppd" is the "log pointwise predictive density" measuring how well the model fits the data, while p_{WAIC} is the sum of the posterior variance of the log-predictive density and estimates the effective number of parameters and serves as a penalty for the model. The difference between lppd and p_{WAIC} is multiplied by -2 to be on the deviance scale (Vehtari et al., 2017; Chan-Golston et al., 2020). Hence, lower WAIC values indicate preferred models.

Since our densities are predominantly matrix-normal, with $\boldsymbol{\Sigma} \sim \mathcal{IW}(n_T, \boldsymbol{D}_T)$ and $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{R}$ being integrable from the $\mathcal{MNIW}$ and $\mathcal{MNIG}$ respectively, the lppd can be computed analytically for each. Without loss of generality, we address the case with $\boldsymbol{\Sigma} \sim \mathcal{IW}(n_T, \boldsymbol{D}_T)$, with moments computed with Algorithm 5: $\mathbb{E}_{\boldsymbol{\Theta}_t|\boldsymbol{Y}_{1:T}}\left[p(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t)\right] = p(\boldsymbol{Y}_t^* \mid \boldsymbol{Y}_{1:T}) = \mathcal{HT}(\boldsymbol{Y}_t^* \mid \boldsymbol{F}_t\boldsymbol{h}_t, \boldsymbol{F}_t\boldsymbol{H}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t, n_T, \boldsymbol{D}_T)$. The log-density follows from the expression for the Hyper-T in equation (5):

$$\begin{aligned}\text{lppd} = &-\frac{NST}{2}\log\pi + \sum_{t=1}^{T}\left\{\log\Gamma_S\left(\frac{n_T+N}{2}\right) - \log\Gamma_S\left(\frac{n_T}{2}\right)\right. \\ &-\frac{S}{2}\log\det(\boldsymbol{F}_t\boldsymbol{H}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t) - \frac{N}{2}\log\det(\boldsymbol{D}_T) \\ &\left.-\frac{n_T+N}{2}\log\det\left(\boldsymbol{I}_S + \boldsymbol{D}_T^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}\boldsymbol{h}_t)^\top(\boldsymbol{F}_t\boldsymbol{H}_t\boldsymbol{F}_t^\top + \boldsymbol{V}_t)^{-1}(\boldsymbol{Y}_t - \boldsymbol{F}_t\boldsymbol{h}_t)\right)\right\}\end{aligned} \tag{18}$$

Sampling is required to compute the p_{WAIC}, with L samples of $\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T}$ taken to get the empirical p_{WAIC}: $\hat{p}_{\text{WAIC}} = \sum_{t=1}^{T} \text{Var}\left[\left\{\log p\left(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t^{(l)}\right)\right\}_{l=1,\dots,L}\right]$.

Patterning after (Vehtari et al., 2017), we compute the standard error of the WAIC by taking the sample variance across time. Denote the term-wise WAIC by the following:

$$\text{WAIC}_t = -2\left(\log\mathbb{E}_{\boldsymbol{\Theta}_t|\boldsymbol{Y}_{1:T}}[p(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t)] - \text{Var}_{\boldsymbol{\Theta}_t|\boldsymbol{Y}_{1:T}}[\log p(\boldsymbol{Y}_t^* \mid \boldsymbol{\Theta}_t)]\right) \tag{19}$$

Then the standard error of the WAIC can be computed as follows:

$$\text{se}(\text{WAIC}) = \sqrt{T\text{Var}\left(\{\text{WAIC}_t\}_{t=1,\dots,T}\right)} \tag{20}$$

3.2 Posterior predictive loss criteria

This criteria based on (Gelfand and Ghosh, 1998) involves sampling independent replicates of the model outcome based on the distribution's computed moments. Denote one such set of replicates by $\boldsymbol{Y}_t^{(\text{rep})}$, where $\boldsymbol{Y}_t^{(\text{rep})}$ is a "future" observation which is replicated from the distribution of $\boldsymbol{Y}_t$. However, since we account for the actual observations $\boldsymbol{Y}_{1,\dots,T}$ in our criteria, the replicate we sample is actually $\boldsymbol{Y}_t^{(\text{rep})} \mid \boldsymbol{Y}_{1,\dots,T}$. For L such replicates, we sample $\{\boldsymbol{Y}_t^{(\text{rep}),l}\}_{l=1,\dots,L}$. Then the sample mean of the L samples is $\hat{\boldsymbol{\mu}}_t^{(\text{rep}),L} = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{Y}_t^{(\text{rep}),l}$ and its per-coordinate variance is $\hat{\sigma}_t^2(\boldsymbol{x}_i, \boldsymbol{s}_j) = \text{Var}((y_t(\boldsymbol{x}_i, \boldsymbol{s}_j)^{(\text{rep}),l})_{l=1,\dots,L})$. These two moments are used to define a D-score Gelfand and Ghosh (1998) as the sum of a goodness of fit measure (G) and a penalization term (P). Here, $D = G + P$, where G and P are defined as

$$G = \sum_t\sum_{i,j}(y_t(\boldsymbol{x}_i, \boldsymbol{s}_j) - \hat{\mu}_t^{(\text{rep}),\text{L}}(\boldsymbol{x}_i, \boldsymbol{s}_j))^2; \qquad P = \sum_t\sum_{i,j}\hat{\sigma}_t^2(\boldsymbol{x}_i, \boldsymbol{s}_j) \tag{21}$$

Due to the normality inherent in the DLM, both of our moments, and hence the G and P scores, can be computed analytically without requiring samples. Hence, $\boldsymbol{\mu}_t^{(\text{rep})} = \mathbb{E}_{\boldsymbol{Y}_t^{(\text{rep})}|\boldsymbol{Y}_{1,\ldots,T}} \hat{\boldsymbol{\mu}}_t^{(\text{rep}),L} = \boldsymbol{F}_t \boldsymbol{h}_t$ (the values of G do not depend on the covariance structure $\boldsymbol{\Sigma}$), while $\sigma_t^2(\boldsymbol{x}_i, \boldsymbol{s}_j) = \mathbb{E}_{\boldsymbol{Y}_t^{(\text{rep})}|\boldsymbol{Y}_{1,\ldots,T}} \hat{\sigma}_t^2(\boldsymbol{x}_i, \boldsymbol{s}_j)$ takes the following forms for each model:

$$\sigma^2_{t,\text{IW}}(\boldsymbol{x}_i, \boldsymbol{s}_j) = \frac{H_{t,ii} D_{T,jj}}{n_T - 2}; \qquad \sigma^2_{t,\text{IG}}(\boldsymbol{x}_i, \boldsymbol{s}_j) = \frac{H_{t,ii} d_T R_{jj}}{n_T - 1}; \qquad \sigma^2_{t,\text{id}}(\boldsymbol{x}_i, \boldsymbol{s}_j) = H_{t,ii}; \quad (22)$$

where $\sigma^2_{t,\text{IW}}$, $\sigma^2_{t,\text{IG}}$ and $\sigma^2_{t,\text{id}}$ denote individual terms in P corresponding to $\boldsymbol{\Sigma}$ being inverse-Wishart, $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{R}$ and $\boldsymbol{\Sigma} = \boldsymbol{I}$, respectively. Summed over the N inputs $\boldsymbol{x}_i$ and S spatial coordinates $\boldsymbol{s}_j$, we write $G = \sum_t ||\boldsymbol{Y}_t - \boldsymbol{\mu}_t^{(rep)}||_F^2$ ($\|\cdot\|_F$ is the Frobenius norm) and P as

$$P_{\text{IW}} = \frac{\text{tr}(\boldsymbol{D}_T)}{n_T - 2} \sum_t \text{tr}(\boldsymbol{H}_t); \qquad P_{\text{IG}} = \frac{d_T \text{tr}(\boldsymbol{R})}{n_T - 1} \sum_t \text{tr}(\boldsymbol{H}_t); \qquad P_{\text{id}} = S \sum_t \text{tr}(\boldsymbol{H}_t) \quad (23)$$

As in (22), each of the terms $P_\cdot$ in (23) denotes the P for the model corresponding to the variance structure $\boldsymbol{\Sigma}$. The lower the D-score of the distribution, the better the fit of the data to the model. Table 2 shows the results of this statistic for the FFBS applied to the data.

4. A Bayesian Transfer Learning Approach for BIG DATA settings

The problem of emulating high-dimensional computer model outputs has been comprehensively addressed by Higdon et al. (2008); Gramacy and Apley (2015); Gu and Berger (2016); Sauer et al. (2023a). Most of these developments have revolved around models that achieve dimension reduction using scalable derivations of Gaussian processes. Even a cursory review reveals a significant literature on statistical methods for massive spatial datasets, which is too vast to be summarized here (see, e.g., Banerjee, 2017; Heaton et al., 2019, and references therein). Within the Bayesian setting, inference proceeds from spatial processes that scale massive data sets. Examples range from reduced-rank processes or subsets of regression approaches (Quinonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005; Cressie and Johannesson, 2008; Banerjee et al., 2008; Wikle, 2010; Wilson and Nickisch, 2015, and references therein), multi-resolution approaches (Nychka et al., 2015; Katzfuss, 2017), and graph-based models inducing sparsity (Datta et al., 2016a; Katzfuss and Guinness, 2021; Krock et al., 2023; Dey et al., 2022; Sauer et al., 2023a; Cao et al., 2023).

We depart from the above model-based approaches and adopt a Bayesian transfer learning approach that will divide and conquer a dataset of possibly massive dimensions, construct a sequence of smaller datasets, and then apply the FFBS algorithm described in Algorithm 5 to this sequence. In spirit, this is similar to spatial meta-kriging or predictive stacking (see Guhaniyogi et al., 2017; Guhaniyogi and Banerjee, 2018; Presicce and Banerjee, 2024, and references therein). To elucidate further, let us use the analogy of a streaming show where the data at each $t = 1, \ldots, T$ is referred to as a "season" for the show and each season consists of K subsets of the data, which we refer to as "episodes". This scheme is depicted in Figure 2, where the data (season) at time t is partitioned into K subsets (episodes) denoted by $\mathcal{D}_{1t}, \ldots, \mathcal{D}_{Kt}$. Each $\mathcal{D}_{kt} = \{\boldsymbol{Y}_{k,t}, \boldsymbol{F}_{k,t}, \boldsymbol{G}_{k,t}, \boldsymbol{V}_{k,t}, \boldsymbol{W}_{k,t}\}$, where each $\boldsymbol{Y}_{k,t}$ is $r \times c$, $\boldsymbol{F}_{k,t}$ is $r \times p$, $\boldsymbol{G}_{k,t}$ is $p \times p$, $\boldsymbol{V}_{k,t}$ and $\boldsymbol{W}_{k,t}$ are positive definite covariance matrices of dimension

$r \times r$ and $p \times p$, respectively. One convenient specification emerges naturally by treating each of the K matrices $\boldsymbol{Y}_{k,t}$ as submatrices of $\boldsymbol{Y}_t$. We let $K = k_1 k_2$, where $rk_1 = N$ and $ck_2 = S$ where we choose r and c to ensure that we can execute Algorithm 5 on each $\mathcal{D}_{k,t}$ as a sequence of datasets over KT time points.

The preceding divide and conquer method conveniently retains the conjugate distribution theory underlying Algorithm 5. The transfer learning mechanism scales the emulation of mechanistic systems when either N or S is large. Specifying $\boldsymbol{V}_{k,t}$ using Gaussian processes, as described in Section 2.2, allows interpolation of $\boldsymbol{Y}_{k,t}(\tilde{\boldsymbol{\chi}})$ for arbitrary units $\tilde{\boldsymbol{\chi}}$. However, inference on the complete spatial field involving S spatial locations is precluded as the $\boldsymbol{\Sigma}$ matrix only captures the dependence among the c columns of $\boldsymbol{Y}_{k,t}$ in each episode. If full inference on the spatial field is desired, then we model the S columns of the full dataset as a spatial covariance matrix from which we extract the $c \times c$ matrix corresponding to each episode in the transfer learning algorithm. More specifically, we consider the model

$$\begin{aligned} \boldsymbol{Y}_{k,t} &= \boldsymbol{F}_{k,t}\boldsymbol{\Theta}_{k,t} + \boldsymbol{\mathcal{E}}_{k,t}, \quad \boldsymbol{\mathcal{E}}_{k,t} \overset{\text{ind.}}{\sim} \mathcal{MN}_{r\times c}(\boldsymbol{O}, \boldsymbol{V}_{k,t}, \boldsymbol{\Sigma}) \\ \boldsymbol{\Theta}_{k,t} &= \boldsymbol{G}_{k,t}\boldsymbol{\Theta}_{\text{pa}[k,t]} + \boldsymbol{\Gamma}_{k,t}, \quad \boldsymbol{\Gamma}_{k,t} \overset{\text{ind.}}{\sim} \mathcal{MN}_{p\times c}(\boldsymbol{O}, \boldsymbol{W}_{k,t}, \boldsymbol{\Sigma}), \end{aligned} \tag{24}$$

where the indices $\{k, t\}$ refer to the k-th episode in season t, $\boldsymbol{V}_{k,t}$ and $\boldsymbol{W}_{k,t}$ are matrices with rows and columns corresponding to the episode k within season t and $\text{pa}[k, t]$ denotes the index that immediately precedes (is the "parent" of) the index $\{k, t\}$ and defined as

$$\text{pa}[k,t] = \begin{cases} \{k-1, t\} \text{ if } k = 2, \ldots, K\,; \quad t = 1, 2, \ldots, T \\ \{K, t-1\} \text{ if } k = 1\,; \quad t = 2, \ldots, T \\ \{0, 0\} \text{ if } k = 1, t = 1\,. \end{cases}.$$

Algorithm 5 requires that we maintain a shared covariance matrix among the columns of $\boldsymbol{Y}_{k,t}$ for each k and t, which is possible only if each episode has the same number of columns. Since each $Y_{k,t}$ is $r \times c$, it follows that the $c \times c$ covariance matrix $\boldsymbol{\Sigma}$ models the covariance among the columns for every episode in the entire stream.

To analyze the computational complexity of Algorithm 5 under the transfer learning framework, we replace the number of locations with c and the number of time points with $K \times T$. This substitution yields a computational cost of $\sim O(KTLc^3)$. The primary benefit of transfer learning in this context is the reduction of the cubic term in complexity from S^3 to Kc^3, which can substantially alleviate the computational burden.

5. Applications for Emulation

5.1 Predator-prey analysis

We apply our methodology to 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. To motivate the ecological computer model, we overview the Lotka-Volterra equations, which models the interaction between the

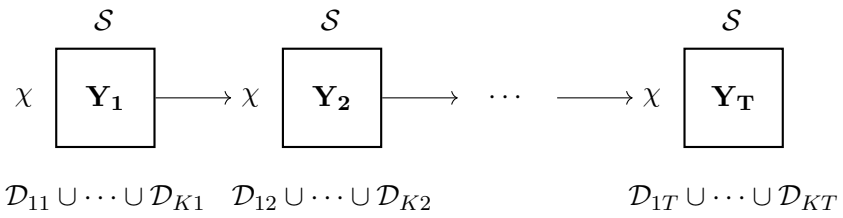

Figure 2: The transfer learning scenario. At each time point, a $\chi \times S$ matrix is recorded. Within each Y_t, information propagates from subdomains within the grid $\mathcal{D}_{1t} \cup \cdots \cup \mathcal{D}_{Kt}$, before propagating into subdomains for the next matrix Y_{t+1}.

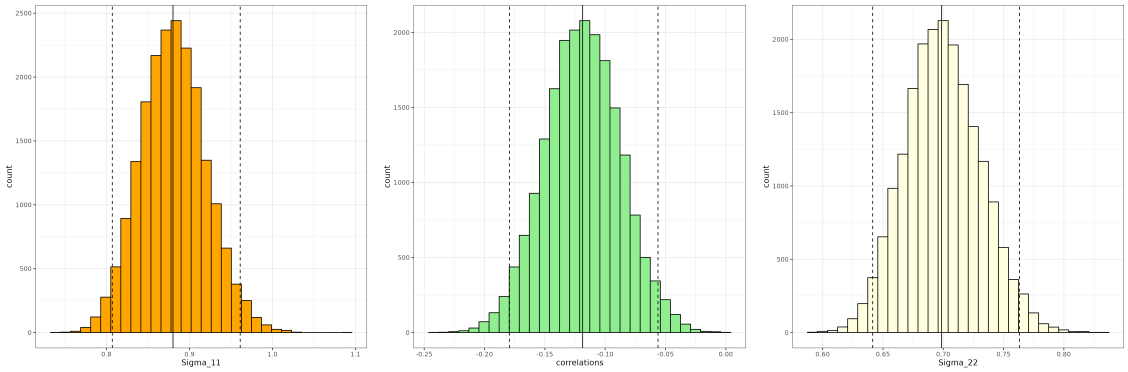

Figure 3: Posterior distributions of the variances of the prey (left), the predator (right), and the correlation between the two populations (center). The 95% credible interval of the correlation is $(-0.179, -0.056)$, capturing the negative relation between the (log-transformed) predator and prey populations.

populations of a single predator species and a single prey species:

$$\begin{aligned} \frac{du_t}{dt} &= \eta_1 u_t - \eta_2 u_t v_t \\ \frac{dv_t}{dt} &= -\eta_3 v_t + \eta_4 u_t v_t \ , \end{aligned} \tag{25}$$

where η_1 is the growth rate of the prey population and η_3 is the rate of population loss within the predator population independent of the interactions the two populations would have with one another; η_2 and η_4 represent the respective rates at which the prey population shrinks and the predator population grows, which is represented through the product of their respective populations. The parameters η_1, η_2, η_3, and η_4 must be positive; furthermore, the values of $\eta_1, \ldots, \eta_4$ must correspond to the units of the quantities they multiply. Thus, η_2 and η_4, in particular, should be one or two orders of magnitude smaller than η_1 and η_3, since they correspond to the product of the populations of the predators and prey.

To train the FFBS model using Algorithm 5, we generate $N = 50$ sets of lognormal sampled parameters, so that $\eta_i \sim \exp\left(\mathcal{N}(\mu_i, \sigma_i)\right)$ for $i = 1, \ldots, 4$, where μ_i and σ_i are

the respective means and standard deviations of the normal random variables prior to exponentiation. The training data is generated over a Latin hypercube design from the multivariate lognormal with $\mu_1 = \mu_3 = 0$ and $\mu_2 = \mu_4 = -3$, and $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.5$, so that η_1 and η_3 will be close to 1 and η_2 and η_4, which control the prey's population decline and the predator's population growth respectively, will be close to 0.05.

We let $\boldsymbol{Y}_t$ be the $N \times 2$ matrix, where row i corresponds to the log-transformed solutions $[u_t, v_t]$ of (25) for the ith training parameter $\boldsymbol{\eta}_i$, $i = 1, \ldots, N$. The design matrix for the model approximation is the data matrix from the prior time point in an AR(1) setup, so that $\boldsymbol{F}_t = \boldsymbol{Y}_{t-1}$ for $t = 1, \ldots, T$ (see, e.g., West and Harrison, 1997, for details of such formulations). We set $\boldsymbol{Y}_0$ to the log-population of the Canadian lynx from the data in the year 1900 and emulate $\boldsymbol{Y}_t$ from (25) over $T = 20$ time points. We also generate $\boldsymbol{V}_t = (C(\boldsymbol{\eta}_i; \boldsymbol{\eta}_j; \beta))$ using $\beta_i = \beta$ in (13) with $\beta = \frac{3}{0.5 \times d_{\max}}$, where $d_{\max}$ is the maximum distance between $\boldsymbol{\eta}_i$'s, and we let $\boldsymbol{G}_t$ and $\boldsymbol{W}_t$ be identity matrices.

Also of interest in fitting Algorithm 5 to the data is a measure for the covariance between the log-predator and log-prey populations, which enables us to compute a correlation measure to interpret the results we have been given. Figure 3 presents the posterior distributions using 20,000 posterior samples of $\boldsymbol{\Sigma}$. The left and right figures are the posterior samples of the diagonal elements, Σ_{11} and Σ_{22}, of $\boldsymbol{\Sigma}$ while the middle panel is the correlation $\Sigma_{12}/\sqrt{\Sigma_{11}\Sigma_{22}}$. The 95% credible interval for the correlation is $(-0.179, -0.056)$, which complies with the dynamics of the predator and prey populations in the Lotka-Volterra cycle. As the prey population increases, the predator population increases at the expense of the prey population, enough to cause the prey to decrease dramatically. Once the prey become scarce, the predators also begin to die off when there are less prey to eat, allowing for the prey population to grow, after which the cycle repeats.

Analytic solutions with their credible intervals are sampled and plotted in Figure 4. The bottom two plots feature the comparison of log-population values between the actual values of the log-prey and log-predator populations and their estimates produced by the FFBS emulator respectively. The 95% credible intervals accurately capture the trajectories of the predator and prey populations; particularly for the middle and right plots.

5.2 Nonlinear Partial Differential Equation Emulation

We apply our methodology to a spatiotemporal model arising from a set of coupled nonlinear PDEs, a natural way to describe the dynamics of a continuous outcome across space and time. Our model uses the set of coupled PDEs shown in (26). This mechanistic system is 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 focuses on proving various mathematical properties with no discussion about parameter inference or connections to real-world data or field measurements. Specifically, Zhang and Wang (2014) posit the existence of traveling waves in an influenza outbreak for a system such as (26), but omit discussion of parameter inference. See Figure 6 for a visualization of this traveling wave phenomenon 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. In this model, disease transmission is predominantly a localized

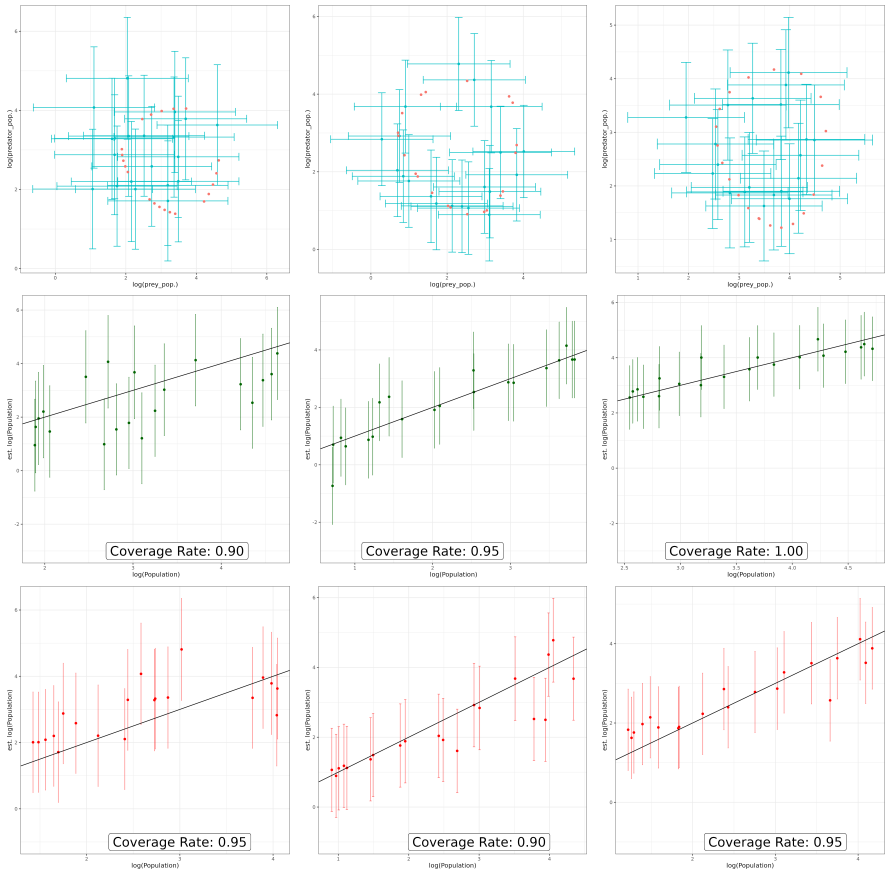

Figure 4: Prey and predator log-populations are plotted against their predicted values with 95% credible intervals for data corresponding to inputs held out of the training phase. The top three plots contain the point estimates with their credible intervals (turquoise) with the ground truth curves in red. The middle three plots correspond to the log-prey populations and the bottom three correspond to the log-predator populations. The black diagonal line denotes the region where the predicted and true values are equal. Each column corresponds to a distinct value of $\boldsymbol{\eta}$.

process where transmission is most likely between nearby locations. The movement of individuals then facilitates the geographical spread of infectious diseases. Such a process may be captured through the following system

$$\begin{aligned}
\frac{\partial S_t(\boldsymbol{s})}{\partial t} &= -\eta_1 \frac{S_t(\boldsymbol{s}) I_t(\boldsymbol{s})}{N} + \alpha_1 \nabla^2 S_t(\boldsymbol{s}) \\
\frac{\partial I_t(\boldsymbol{s})}{\partial t} &= \eta_1 \frac{S_t(\boldsymbol{s}) I_t(\boldsymbol{s})}{N} - \eta_2 I_t(\boldsymbol{s}) + \alpha_2 \nabla^2 I_t(\boldsymbol{s}) \\
\frac{\partial R_t(\boldsymbol{s})}{\partial t} &= \eta_2 I_t(\boldsymbol{s}) + \alpha_3 \nabla^2 R_t(\boldsymbol{s}),
\end{aligned} \tag{26}$$

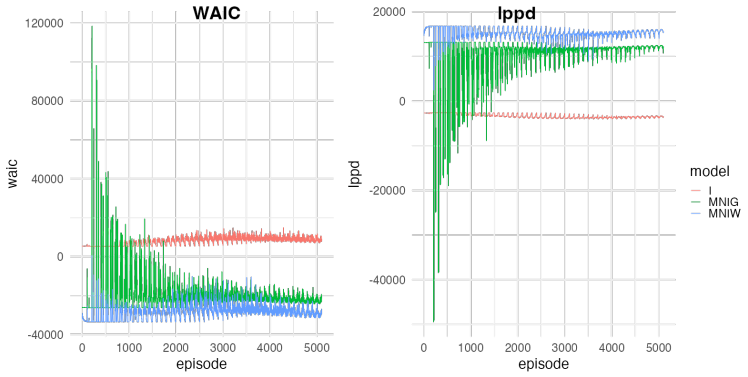

Figure 5: WAIC over episode/time for FFBS models with three different variance structures: inverse-Wishart, inverse-Gamma with spatial correlation matrix $\boldsymbol{R}$, and $\boldsymbol{I}$.

where $\nabla^2 f_t(\boldsymbol{s}) = \sum_{i=1}^{d} \partial^2 f_t(\boldsymbol{s})/\partial \boldsymbol{s}_i^2$. In applying our methodology, the training data is generated through the numerical solution of (26) for fixed parameter setting $\boldsymbol{x} = (\eta_1', \eta_2', \alpha_1', \alpha_2', \alpha_3')$. The outcome $y_t(\boldsymbol{s}, \boldsymbol{x})$ denotes the infection counts at a location and time, namely $I_t(\boldsymbol{s})$.

We employ `deSolve` (`R` package version 1.40) (Soetaert et al., 2010) to solve the system over the range of input parameters. Our spatial domain consists of a 100×100 grid, resulting in $S = 10,000$ spatial locations. The system's initial state begins with two seeding locations: one in the middle and one in the bottom left corner. We generated $N = 50$ sets of $\boldsymbol{x}$ used in equation (26), and for each input generated solutions over $T = 50$ time points from the PDE system, resulting in 51 time points in total including time 0.

Letting the number of infected people at time t over N inputs and S spatial coordinates be the $N \times S$ matrix $\mathbb{I}_t$, we set the data matrix $\boldsymbol{Y}_t = \log(\mathbb{I}_t + 1)$. We further adopt a second-order autoregressive (AR2) structure for the covariate matrix $\boldsymbol{F}_t$, but structure it so that each episode has access to data from the previous two seasons, so that $\boldsymbol{F}_{k,t} = [\boldsymbol{Y}_{k,t-1}, \boldsymbol{Y}_{k,t-2}]$. As with the predator-prey example in the preceding section, we also specify $\boldsymbol{V}_{k,t} = (C(\boldsymbol{\eta}_i, \boldsymbol{\eta}_j; \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is defined the same way, as well as $\boldsymbol{G}_t$ and $\boldsymbol{W}_t$ as the identity matrices. We use 40 generated parameters to train the FFBS algorithm (Algorithm 5), taking $L = 10$ samples in the backward sampling step, and employ a Gaussian Process to emulate the PDE solution. We use the transfer learning parameters specified in Section 4 set as $r = 40$ and $c = 100$, so that we regress on one column of the grid at each step to efficiently emulate our PDE.

We emulate the space-time field of the PDE using three different variance structures: (i) $\boldsymbol{\Sigma}$ follows inverse-Wishart; (ii) $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{R}$; and (iii) $\boldsymbol{\Sigma} = \boldsymbol{I}$. Table 1 shows the WAIC and its component statistics. The inverse-Wishart model outperforms the inverse-Gamma and identity covariance. Figure 5 shows the details of WAIC over episode/time for FFBS models with three different variance structures. Table 2 presents the GPD scores ($D = G + P$ from (21)) obtained from independent replicates. The inverse-Wishart model performs slightly better than the inverse-Gamma model, and both outperform the identity model.

Figure 6 shows an example of the dynamics of Equation (26) generated by deSolve. The disease spreads over time to cover nearly the entire grid at $t = 37$, but the people closest to the source points have already begun to recover. By the final time point of the simulation

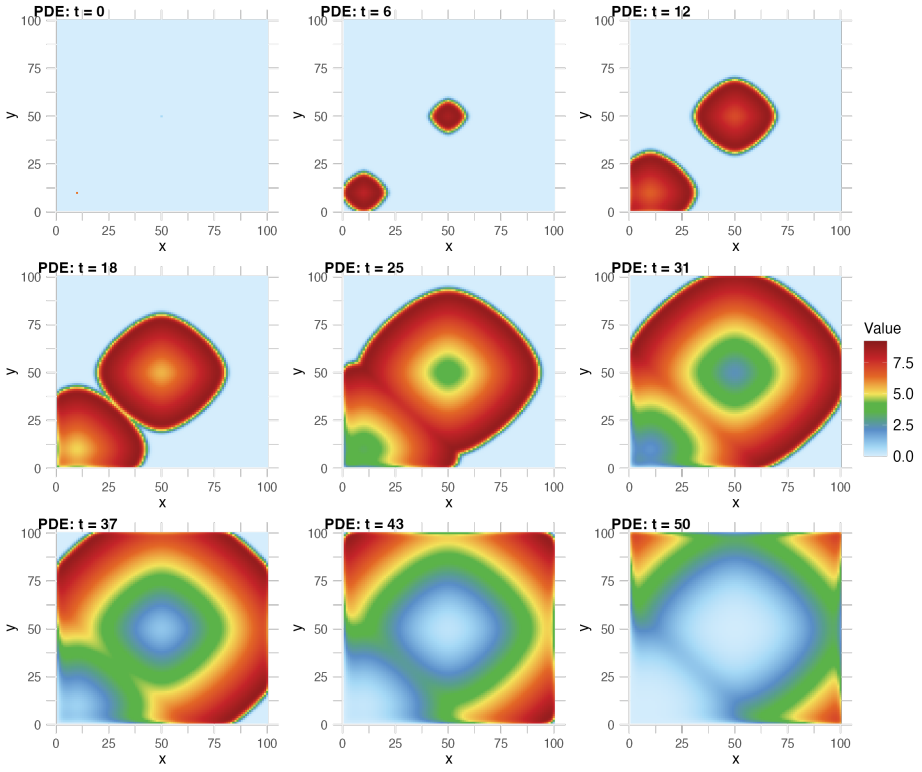

Figure 6: Spatiotemporal dynamics of coupled nonlinear partial differential equations. Darker colors indicate larger function values. Wave-like dynamics radiating outwards from the initial seeding locations are apparent.

$t = 50$, nearly everyone in the grid has recovered from the disease, with its prevalence being restricted to the corners of the grid, which caught the disease later than the other points.

Figure 7 compares the PDE solution and the FFBS emulation results using the $\mathcal{MNIW}$ covariance structure at the same selected time points. The boundaries and overall region of the spread of the disease is accurately emulated, with some inaccuracies taking place within the infected regions. By $t = 31$, however, even the number of infected people within the infected region has been accurately emulated, and by $t = 43$, the emulation is nearly perfect as revealed by comparing the fields in the second row of Figure 7 with those in the first row (also corresponding to $t = 18, 31, 43$ in Figure 6).

Figure 8 contrasts the PDE solution and FFBS emulated values across all locations, with the same parameters as in Figure 7. Both figures demonstrate good predictive accuracy of the FFBS algorithm in emulating the nonlinear PDE.

6. Estimating mechanistic system parameters using field data

The above emulator is now melded with the field observations to infer about the mechanistic parameters. Since the mechanistic system may not adequately describe observed field data

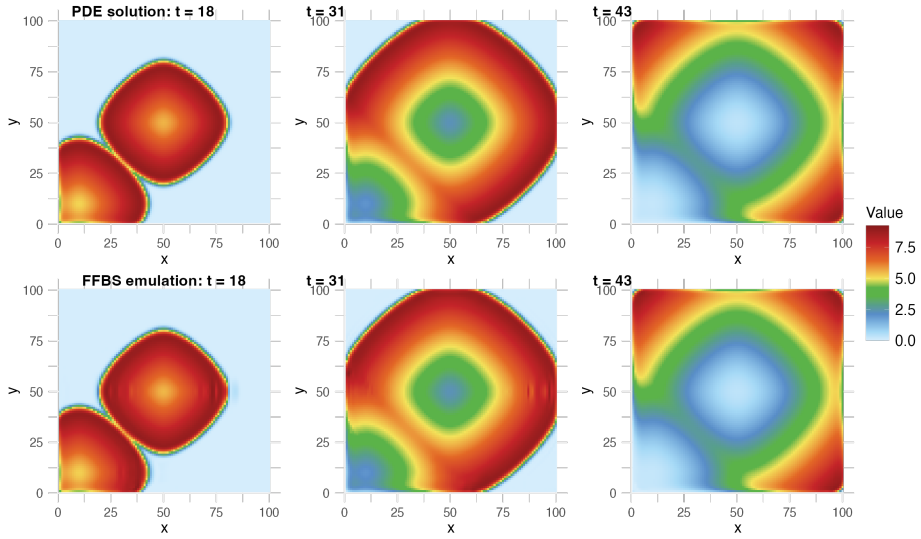

Figure 7: Heatmap comparison of one example of spatiotemporal fields between PDE solutions and FFBS emulations. The parameters are $\eta_1 = 3.549$, $\eta_2 = 0.268$, $\alpha_1 = 0.010$, $\alpha_2 = 0.143$, and $\alpha_3 = 0.170$ at times $t = 18, 31, 43$. The first row presents the space-time fields generated by the PDE, while the second row estimates the field for some unobserved inputs held out of the training data.

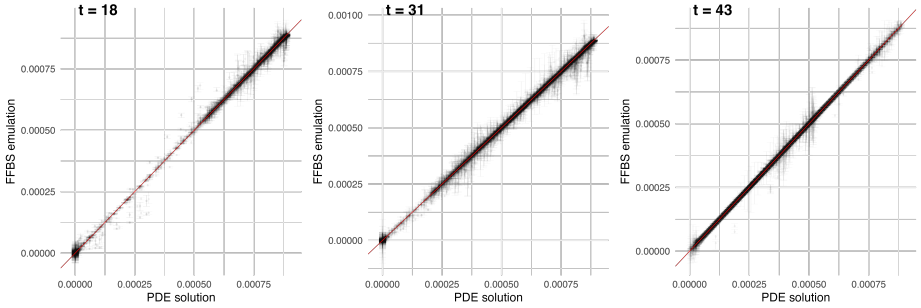

Figure 8: Scatter plot featuring a 95% credible interval that contrasts PDE solutions and FFBS emulation results corresponding to some unobserved inputs held out of training data. The red line denotes the region where emulation is perfect.

over all spatial locations and time points, we include a spatiotemporally-varying discrepancy term that captures the bias between the observed field data and emulator. Our state-space model with dynamic bias correction $\boldsymbol{u}$ is

$$\begin{aligned} z_t(\boldsymbol{s}) &= y_t(\boldsymbol{\eta}, \boldsymbol{s}) + u_t(\boldsymbol{s}) + \varepsilon_t^z(\boldsymbol{s}), \quad \varepsilon_t^z(\boldsymbol{s}) \overset{\text{ind.}}{\sim} \mathcal{N}(0, \tau_t^2) \\ u_t(\boldsymbol{s}) &= u_{t-1}(\boldsymbol{s}) + \varepsilon_t^u(\boldsymbol{s}), \quad \varepsilon_t^u(\boldsymbol{s}) \overset{\text{ind.}}{\sim} \mathcal{GP}(0, \tau_t^2 C(\cdot, \cdot; \boldsymbol{\rho})) \\ \tau_t^2 &\overset{\text{ind.}}{\sim} \mathcal{IG}(b^t n_0^z, b^t d_0^z), \quad t = 1, 2, \ldots, T \end{aligned} \tag{27}$$

	$\boldsymbol{\Sigma}$ = inverse-Wishart	$\boldsymbol{\Sigma} = \sigma^2\boldsymbol{R}$	$\boldsymbol{\Sigma} = \boldsymbol{I}$ (reference)
lppd	77.8×10^6	55.7×10^6	-17.1×10^6
lppd (analytic computation)	79.3×10^6	57.1×10^6	-16.2×10^6
p_{WAIC}	5.2×10^6	4.7×10^6	3.2×10^6
WAIC $= -2(\text{lppd} - p_{\text{WAIC}})$	-145.4×10^6	-101.9×10^6	40.6×10^6

Table 1: The WAIC and its component statistics computed from the PDE emulation data with different variance structure $\boldsymbol{\Sigma}$ including inverse-Wishart, $\sigma^2\boldsymbol{R}$, and $\boldsymbol{I}$.

	$\boldsymbol{\Sigma}$ = inverse-Wishart	$\boldsymbol{\Sigma} = \sigma^2\boldsymbol{R}$	$\boldsymbol{\Sigma} = \boldsymbol{I}$ (reference)
G	3.0×10^4	3.0×10^4	3.0×10^4
P	1.9×10^7	3.0×10^7	935.0×10^7
$D = G + P$	1.9×10^7	3.0×10^7	935.0×10^7

Table 2: The model comparison with independent replicates. Mirroring the results from Table 1, the inverse-Wishart $\boldsymbol{\Sigma}$ has the best fit, followed by $\boldsymbol{\Sigma} = \sigma^2\boldsymbol{R}$, and then by $\boldsymbol{\Sigma} = \boldsymbol{I}$.

where $y_t(\boldsymbol{\eta}, \boldsymbol{s})$ denotes the emulator prediction at mechanistic input $\boldsymbol{\eta}$ and location $\boldsymbol{s}$. We assume that the field data are available over a set of $\tilde{S}$ spatial locations in a set $\tilde{\mathcal{S}} \subseteq \mathcal{S}$. It is worth pointing out that in the traditional exercise of calibrating computer models, the discrepancy term is treated as function of the model inputs. One could certainly incorporate $u_t(\boldsymbol{\eta}, \boldsymbol{s})$ in (27). However, such paradigms assume that the field data are partial realizations of a process with some unknown "optimal" η that is adjusted by the discrepancy terms and the noise. We do not need such assumptions for inferring about $\boldsymbol{\eta}$. Instead, we treat the problem as one of a mixed nonlinear regression with the field observations regressed by the mechanistic output for some unknown $\boldsymbol{\eta}$. The space-time discrepancy term now serves as a process that attempts to learn from the mechanistic system and the data.

Let $\boldsymbol{u}_t$ be the $\tilde{S} \times 1$ vector with elements $u_t(\boldsymbol{s}_i)$ and $\boldsymbol{U}(\boldsymbol{\rho})$ be the $\tilde{S} \times \tilde{S}$ correlation matrix built using $C(\cdot, \cdot; \boldsymbol{\rho})$. The posterior distribution conditional on the collection of field observations, $\boldsymbol{z}_{1:T}$, and the emulated values, $\boldsymbol{Y}_{1:T}$, is proportional to the joint distribution

$$\begin{aligned} p(\boldsymbol{\eta}, \boldsymbol{\rho}) \times \mathcal{NG}(\boldsymbol{u}_0, \tau_0^{-2} | \boldsymbol{m}_0^z, \boldsymbol{M}_0^z, n_0^z, d_0^z) \times \prod_{t=1}^{T} \mathcal{NG}(\boldsymbol{u}_t, \tau_t^{-2} \mid \boldsymbol{u}_{t-1}, \boldsymbol{U}(\boldsymbol{\rho}), b^t n_0^z, b^t d_0^z) \\ \times \prod_{t=1}^{T} \left\{ \mathcal{N}(\boldsymbol{y}_t(\boldsymbol{\eta}) \mid \tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta}), \tilde{\boldsymbol{\Sigma}}(\boldsymbol{\eta})) \prod_{i=1}^{\tilde{S}} \mathcal{N}\left(z_t(\boldsymbol{s}_i) \mid y_t(\boldsymbol{\eta}, \boldsymbol{s}_i) + u_t(\boldsymbol{s}_i), \tau_t^2\right) \right\}, \end{aligned} \tag{28}$$

where $\boldsymbol{y}_t(\boldsymbol{\eta})$ is the $\tilde{S} \times 1$ vector with elements $y_t(\boldsymbol{\eta}, \boldsymbol{s})$ for each location $\boldsymbol{s} \in \tilde{\mathcal{S}}$, $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta})^\top = \tilde{\boldsymbol{F}}_t(\boldsymbol{\eta})\boldsymbol{\Theta}_t(\tilde{\mathcal{S}}) + \boldsymbol{J}_t(\boldsymbol{\eta})^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t(\tilde{\mathcal{S}}) - \boldsymbol{F}_t\boldsymbol{\Theta}_t(\tilde{\mathcal{S}}))$ and $\tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta}) = (1 - \boldsymbol{J}_t(\boldsymbol{\eta})^\top \boldsymbol{V}_t^{-1} \boldsymbol{J}_t(\boldsymbol{\eta}))\boldsymbol{\Sigma}(\tilde{\mathcal{S}})$ are the $1 \times \tilde{S}$ mean vector and $\tilde{S} \times \tilde{S}$ covariance matrix, respectively, derived from the conditional predictive distribution in (11) for $\boldsymbol{y}_t(\boldsymbol{\eta})$ given the emulated values $\boldsymbol{Y}_{1:T}$, $\tilde{\boldsymbol{F}}(\boldsymbol{\eta})$ is $1 \times p$ with entries equal to the row in $\tilde{\boldsymbol{F}}_t$ corresponding to $\boldsymbol{\eta}$, $\boldsymbol{\Theta}_t(\tilde{\mathcal{S}})$ are $\boldsymbol{Y}_t(\tilde{\mathcal{S}})$ are $p \times \tilde{S}$ and $N \times \tilde{\mathcal{S}}$ consisting of columns corresponding to locations in $\tilde{\mathcal{S}}$ extracted from $\boldsymbol{\Theta}_t$ and $\boldsymbol{Y}_t$, respectively, $\boldsymbol{J}_t(\boldsymbol{\eta})$ is $N \times 1$ with jth element given by the covariance function $C(\boldsymbol{x}_i, \boldsymbol{\eta}; \boldsymbol{\beta})$ in (13) and

$\boldsymbol{V}_t$ is $N \times N$ as defined in Section 2.2. The definition of $\boldsymbol{\Sigma}(\tilde{\mathcal{S}})$ depends upon our specific choice of the emulation model. For example, if we use (2) for emulation and if the field data locations in $\tilde{\mathcal{S}}$ are a subset of the S locations in $\mathcal{S}$ that were used for emulation, then $\boldsymbol{\Sigma}(\tilde{\mathcal{S}})$ is the $\tilde{S} \times \tilde{S}$ sub-matrix of $\boldsymbol{\Sigma}$ corresponding to the locations in $\tilde{\mathcal{S}}$. Alternatively, if one uses (14) for emulation, then $\boldsymbol{\Sigma}(\tilde{\mathcal{S}}) = \sigma^2 \boldsymbol{R}(\tilde{\mathcal{S}})$, where $\boldsymbol{R}(\tilde{\mathcal{S}})$ is the $\tilde{S} \times \tilde{S}$ spatial correlation matrix formed over the locations in $\tilde{\mathcal{S}}$ using a spatial correlation function; the field data locations need not be a subset of emulator locations.

Algorithm 6 Full conditional distributions for $p(\tau^2_{1:T} \mid \cdot)$.

```
function TauSq_post(u_{0:T}, y_{1:T}, z_{1:T}, n_0, d_0, b, U)
    for t = 1 to T do
        Q_1 ← (u_t − u_{t−1})^⊤ U^{−1} (u_t − u_{t−1}), Q_2 ← (z_t − y_t − u_t)^⊤ (z_t − y_t − u_t)   ▷ O(S̃^2)
        Sample τ_t^2 ~ IG(b^t n_0 + S̃, b^t d_0 + (1/2)(Q_1 + Q_2))
    end for
    return τ^2_{1:T}
end function                                                                                        ▷ O(T S̃^2)
```

We draw samples from $p(\boldsymbol{\eta}, \boldsymbol{\rho}, \tau^2_{0:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}), \boldsymbol{u}_{0:T} | \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T})$ given by

$$\int p(\boldsymbol{\eta}, \boldsymbol{\rho}, \tau^2_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}), \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T}) d\boldsymbol{\Sigma} d\boldsymbol{\Theta}_{0:T}$$
$$= \int p(\boldsymbol{\eta}, \boldsymbol{\rho}, \tau^2_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}) \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T}, \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T}) \times \underbrace{p(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T} \mid \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T})}_{\text{Modularize: } p(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T} | \boldsymbol{Y}_{1:T})} d\boldsymbol{\Sigma} d\boldsymbol{\Theta}_{0:T} \tag{29}$$

where $\boldsymbol{y}_{1:T}(\boldsymbol{\eta}) = (\boldsymbol{y}_1(\boldsymbol{\eta})^\top, \ldots, \boldsymbol{y}_T(\boldsymbol{\eta})^\top)^\top$ is the $\tilde{S}T \times 1$ vector with $\boldsymbol{y}_t(\boldsymbol{\eta})$ as the $\tilde{S} \times 1$ vector with elements $y_t(\boldsymbol{\eta}, \boldsymbol{s}_i)$ for $i = 1, \ldots, \tilde{S}$. A pragmatic approach to sampling from (29) relies on the posterior in (28) and the notion of *Bayesian Modularization* (Bayarri et al., 2009). Modularization replaces the underlined expression in (29) with the lower dimensional distribution $p(\boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T} | \boldsymbol{Y}_{1:T})$, which is readily sampled through FFBS and Metropolis-Hastings steps of Algorithm 11. For each drawn value of $\{\boldsymbol{\Sigma}, \boldsymbol{\Theta}\}$, we will need to draw from $p(\boldsymbol{\eta}, \boldsymbol{\rho}, \tau^2_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}) \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T}, \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T})$.

Drawing samples from $p(\boldsymbol{\eta}, \boldsymbol{\rho}, \tau^2_{0:T}, \boldsymbol{u}_{0:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}) \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{0:T}, \boldsymbol{z}_{1:T}, \boldsymbol{Y}_{1:T})$ employs Gibbs updates using full conditional distributions for $\boldsymbol{y}_{1:T}(\boldsymbol{\eta})$ and $\tau^2_{0:T}$, FFBS updates for $\boldsymbol{u}_{0:T}$, and Metropolis random-walk updates for $\boldsymbol{\eta}$ and $\boldsymbol{\rho}$.

The full conditional distributions for $\tau^2_t \mid \cdot$ are $\mathcal{IG}(n^z_t, d^z_t)$ where $n^z_t = \tilde{S} + b^t n^z_0$ and $d^z_t = b^t d^z_0 + \frac{1}{2}\{(\boldsymbol{u}_t - \boldsymbol{u}_{t-1})^\top \boldsymbol{U}(\boldsymbol{\rho})^{-1}(\boldsymbol{u}_t - \boldsymbol{u}_{t-1}) + \sum_{i=1}^{\tilde{S}} (z_t(\boldsymbol{s}_i) - y_t(\boldsymbol{\eta}, \boldsymbol{s}_i) - u(\boldsymbol{s}_i))^2\}$ for $t = 1, \ldots, T$ and $i = 1, \ldots, \tilde{S}$. The full conditional distributions for $\boldsymbol{y}_t(\boldsymbol{\eta}) \mid \cdot$ are $\mathcal{N}(\boldsymbol{B}_t(\boldsymbol{\eta})\boldsymbol{b}_t(\boldsymbol{\eta}), \boldsymbol{B}_t(\boldsymbol{\eta}))$ where $\boldsymbol{B}_t(\boldsymbol{\eta})^{-1} = \tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta})^{-1} + \tau_t^{-2} \boldsymbol{I}_{\tilde{S}}$ and $\boldsymbol{b}_t(\boldsymbol{\eta}) = \tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta})^{-1} \tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta}) + \tau_t^{-2}(\boldsymbol{z}_t - \boldsymbol{u}_t)$ with $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta}) = \big(\tilde{\mu}_t(\boldsymbol{\eta}, \boldsymbol{s}_1), \ldots, \tilde{\mu}_t(\boldsymbol{\eta}, \boldsymbol{s}_{\tilde{S}})\big)^\top$. Algorithm 6 provides the steps to compute and draw samples from the full conditional distributions of $\tau^2_{1:T} \mid \cdot$ with a computational cost of $O(T\tilde{S}^2)$. Algorithm 7 provides the steps to compute and draw samples from the full conditionals for $\boldsymbol{y}_{1:T} \mid \cdot$ with a computational cost of $O(T\tilde{S}^3)$.

Algorithm 7 Full conditional distributions for $p(\boldsymbol{y}_{1:T} \mid \cdot)$.

function Y_POST($\boldsymbol{\eta}, \tau^2_{1:T}, \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}, \tilde{\mathcal{S}}$)
  for $t = 1$ to T **do**
    $\tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta})^\top \leftarrow \tilde{\boldsymbol{F}}_t(\boldsymbol{\eta})\boldsymbol{\Theta}_t(\tilde{\mathcal{S}}) + \boldsymbol{J}_t(\boldsymbol{\eta})^\top \boldsymbol{V}_t^{-1}(\boldsymbol{Y}_t(\tilde{\mathcal{S}}) - \boldsymbol{F}_t\boldsymbol{\Theta}_t(\tilde{\mathcal{S}}))$ ▷ $O(pN\tilde{\mathcal{S}})$
    $\tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta}) \leftarrow \left(1 - \boldsymbol{J}_t(\boldsymbol{\eta})^\top \boldsymbol{V}_t^{-1}\boldsymbol{J}_t(\boldsymbol{\eta})\right)\boldsymbol{\Sigma}(\tilde{\mathcal{S}})$ ▷ $O(N^2)$
    $\boldsymbol{B}_t(\boldsymbol{\eta}) \leftarrow (\tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta})^{-1} + \tau_t^{-2}\boldsymbol{I}_{\tilde{S}})^{-1}$, $\boldsymbol{b}_t(\boldsymbol{\eta}) \leftarrow \tilde{\boldsymbol{\Sigma}}_t(\boldsymbol{\eta})^{-1}\tilde{\boldsymbol{\mu}}_t(\boldsymbol{\eta}) + \tau_t^{-2}(\boldsymbol{z}_t - \boldsymbol{u}_t)$ ▷ $O(\tilde{\mathcal{S}}^3)$
    Sample $\boldsymbol{y}_t \sim \mathcal{N}(\boldsymbol{B}_t\boldsymbol{b}_t, \boldsymbol{B}_t)$ ▷ $O(\tilde{\mathcal{S}}^3)$
  end for
  return $\boldsymbol{y}_{1:T}, \boldsymbol{B}_{1:T}, \boldsymbol{b}_{1:T}$
end function ▷ $O(T\tilde{\mathcal{S}}^3)$

The model bias process realizations, $\boldsymbol{u}_{0:T}$, are updated from its joint full conditional, $p(\boldsymbol{u}_{0:T} \mid \cdot)$ using the FFBS algorithm. As with emulation, we compute the moments for $\boldsymbol{u}_t \mid \cdot, \boldsymbol{z}_{1:t}, \boldsymbol{y}_{1:t}(\boldsymbol{\eta})$ using the Forward Filter, and then acquire the smoothed moments of $\boldsymbol{u}_t \mid \cdot, \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta})$ using Backwards Sampling. Note that $\boldsymbol{u}_t$ are vectors relevant to Gibbs sampling, we only need one sample of $\boldsymbol{u}_{0:T} \mid \cdot$ per iteration. Algorithm 8 outlines the steps for generating samples of $\boldsymbol{u}_{0:T} \mid \cdot, \boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta})$ using FFBS algorithm. Lines 5-12 describe the Forward Filtering steps, with a computational cost of $\sim O(T\tilde{S}^2)$. Similarly, Lines 14-21 detail the Backward Sampling steps, also with a cost of $\sim O(T\tilde{S}^2)$. Thus, the total computational cost is $\sim O(T\tilde{S}^2)$.

Finally, we consider the density for $\boldsymbol{\rho}, \boldsymbol{\eta} \mid \cdot$, which we glean from the relevant terms in (28). Unfortunately, even when we choose comparatively simple priors for $p(\boldsymbol{\rho}, \boldsymbol{\eta})$ (e.g. multivariate uniform distributions), there is no recognized density that would enable us to sample from known distributions. Consequently, we update $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$ at each iteration via the Metropolis algorithm. Algorithm 10 gives steps for the Metropolis random walk. The target density is given in Algorithm 9, which corresponds to terms in (28).

Since our use cases depend on the entries of $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$ being positive, or at the very least nonnegative, we first transform the parameters to the real line and use a normal density for proposing updates in the Metropolis sampler. Let $\boldsymbol{\phi} = (\boldsymbol{\rho}^\top, \boldsymbol{\eta}^\top)^\top$ be the concatenation of $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$, $l_{\boldsymbol{\phi}}$ be the total number of elements of $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$, $\boldsymbol{\delta} \leftarrow \mathcal{N}(0, \epsilon_3^2\boldsymbol{I})$ denote the proposal update, and $\boldsymbol{g}(\cdot)$ be the function rescaling each element of $\boldsymbol{\phi}$ to the real line. The resulting transformation necessitates the jacobian $\mathcal{J}_{\boldsymbol{\phi}}(\boldsymbol{g}(\boldsymbol{\phi}))$ for both $\boldsymbol{\phi}$ and $\boldsymbol{\phi}^* \leftarrow \boldsymbol{g}^{-1}(\boldsymbol{g}(\boldsymbol{\phi}) + \boldsymbol{\delta})$, with $\boldsymbol{g}(\boldsymbol{\phi})$ and $\boldsymbol{g}(\boldsymbol{\phi}^*)$ considered distributed on $\mathcal{N}(0, 1)$.

We can employ Bayesian modularization to prevent the model discrepancy term $\boldsymbol{u}_t$ from interfering with inference on mechanistic parameters. This additional modularization treats the emulator as an unbiased representation of the field data. After fully sampling the model parameters, each update of the dynamic bias term conditions on a draw from the posterior of the parameters to model the discrepancy between the field data and emulator predictive distribution. Algorithm 11 outlines the steps for learning about the mechanistic system parameters using field data. Each iteration updates the parameters $\tau^2_{1:T}$, $\boldsymbol{u}_{1:T}$, $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$ (via $\boldsymbol{\phi}$), and $\boldsymbol{y}_{1:T}$ using the updated values in order; Line 7 in particular proposes new values using a Metropolis update. Repeating the process for L iterations results in a total computational cost of $O(LT\tilde{\mathcal{S}}^3)$.

Algorithm 8 FFBS for calibration with computer model bias

Input: Field data $\boldsymbol{z}_{1:T}$, emulation results $\boldsymbol{y}_{1:T}(\boldsymbol{\eta})$, starting values $\boldsymbol{m}_0^z$, τ_0^2, $\boldsymbol{M}_0^z$, data scale variance $\tau_{1:T}^2$, and correlation matrix $\boldsymbol{U}$.
Output: Calibration samples $\boldsymbol{u}_{0:T}$
function FFBS_CALIBRATION($\boldsymbol{z}_{1:T}, \boldsymbol{y}_{1:T}(\boldsymbol{\eta}), \boldsymbol{m}_0^z, \tau_0^2\boldsymbol{M}_0^z, \tau_{1:T}^2, \boldsymbol{U}$)
    # Forward Filter
    for $t = 1$ to T **do**
        # Compute prior distribution covariance matrix
        $\tilde{\boldsymbol{A}}_t \leftarrow \tau_{t-1}^2\boldsymbol{M}_{t-1}^z + \tau_t^2\boldsymbol{U}$ ▷ $O(\tilde{S}^2)$
        # Compute one-step ahead forecast covariance matrix
        $\tilde{\boldsymbol{Q}}_t \leftarrow \tilde{\boldsymbol{A}}_t + \tau_t^2\boldsymbol{I}_{\tilde{S}}$ ▷ $O(\tilde{S})$
        # Compute filtering distribution moments
        $\boldsymbol{m}_t^z \leftarrow \boldsymbol{m}_{t-1}^z + \tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{Q}}_t^{-1}(\boldsymbol{z}_t - \boldsymbol{y}_t(\boldsymbol{\eta}) - \boldsymbol{m}_{t-1}^z)$; $\boldsymbol{M}_t^z \leftarrow \tau_t^{-2}(\tilde{\boldsymbol{A}}_t - \tilde{\boldsymbol{A}}_t\tilde{\boldsymbol{Q}}_t^{-1}\tilde{\boldsymbol{A}}_t)$ ▷ $O(\tilde{S}^3)$
    end for
    # Backwards Sampling
    $\tilde{\boldsymbol{h}}_T \leftarrow \boldsymbol{m}_T^z$; $\tilde{\boldsymbol{H}}_T \leftarrow \tau_T^2\boldsymbol{M}_T^z$
    Sample $\boldsymbol{u}_T \sim \mathcal{N}(\tilde{\boldsymbol{h}}_T, \tilde{\boldsymbol{H}}_T)$ ▷ $O(\tilde{S}^3)$
    for $t = T-1$ to 0 **do**
        # Compute BS smoothing distribution moments
        $\tilde{\boldsymbol{h}}_t \leftarrow \boldsymbol{m}_t^z + \tau_t^2\boldsymbol{M}_t^z\tilde{\boldsymbol{A}}_{t+1}^{-1}(\tilde{\boldsymbol{h}}_{t+1} - \boldsymbol{m}_t^z)$; ▷ $O(\tilde{S}^3)$
        $\tilde{\boldsymbol{H}}_t \leftarrow \tau_t^2\boldsymbol{M}_t^z - \tau_t^4\boldsymbol{M}_t^z\tilde{\boldsymbol{A}}_{t+1}^{-1}(\tilde{\boldsymbol{A}}_{t+1} - \tilde{\boldsymbol{H}}_{t+1})\tilde{\boldsymbol{A}}_{t+1}^{-1}\boldsymbol{M}_t^z$ ▷ $O(\tilde{S}^3)$
        Sample $\boldsymbol{u}_t \sim \mathcal{N}(\tilde{\boldsymbol{h}}_t, \tilde{\boldsymbol{H}}_t)$ ▷ $O(\tilde{S}^3)$
    end for
    return $\boldsymbol{u}_{0:T}$
end function ▷ $O(T\tilde{S}^3)$

Algorithm 9 Log probability density of the full conditional $p(\boldsymbol{\phi} \mid \cdot)$.

function LOGLIK($\boldsymbol{\phi} := (\boldsymbol{\rho}, \boldsymbol{\eta}), \tau_{1:T}^2, \boldsymbol{u}_{1:T}, \boldsymbol{z}_{1:T}, \tilde{\mathcal{S}}$)
    $\boldsymbol{y}_{1:T}, \boldsymbol{B}_{1:T}, \boldsymbol{b}_{1:T} \leftarrow$ y_post$\left(\boldsymbol{\eta}, \tau_{1:T}^2, \boldsymbol{z}_{1:T}, \boldsymbol{u}_{1:T}, \tilde{\mathcal{S}}\right)$ ▷ Algorithm 7
    $\mathcal{J}(\boldsymbol{\phi}) \leftarrow \left|\det\left(\frac{\partial g(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}}\right)\right|$
    $\log p(\boldsymbol{\phi} \mid \cdot) = \sum_{t=1}^{T}\Big(\log\mathcal{N}\left(\boldsymbol{y}_t \mid \boldsymbol{B}_t\boldsymbol{b}_t, \boldsymbol{B}_t\right) + \sum_{i=1}^{\tilde{S}}\log\mathcal{N}\left(z_t(\boldsymbol{s}_i) \mid y_t(\boldsymbol{s}_i) + u_t(\boldsymbol{s}_i), \tau_t^2\right)$
        $+ \log\mathcal{N}\left(\boldsymbol{u}_t \mid \boldsymbol{u}_{t-1}, \tau_t^2\boldsymbol{U}(\boldsymbol{\rho})\right)\Big) + \log p(\boldsymbol{\phi}) - \log\mathcal{J}(\boldsymbol{\phi})$
    return $\log p(\boldsymbol{\phi} \mid \cdot)$
end function ▷ $O(T\tilde{\mathcal{S}}^3)$

7. Applications for calibration

7.1 Predator-prey analysis

We estimate parameters in the Lotka-Volterra equations in (25). We fix $\boldsymbol{\rho} = 1.5$ and, analogous to Section 5.1, set the prior for $\boldsymbol{\eta}$ to be a multivariate lognormal. Then $g_i(\eta_i) =$

Algorithm 10 Metropolis random walk algorithm (one step).

function METROP($\boldsymbol{\theta}, p(), \boldsymbol{\Upsilon}$)
  Draw $\boldsymbol{\theta}^* \sim q(\cdot \mid \boldsymbol{\theta}) := N(\cdot \mid \boldsymbol{\theta}, \boldsymbol{\Upsilon})$;
  Set $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ with probability $\alpha = \min\{1, \exp[\log p(\boldsymbol{\theta}^* \mid \cdot) - \log p(\boldsymbol{\theta} \mid \cdot)]\}$.
  return $\boldsymbol{\theta}$
end function

Algorithm 11 Sampler for estimating mechanistic system parameters using field data

Given: Field data $\boldsymbol{z}_{1:T}$, emulation data $\boldsymbol{Y}_{1:T}$, L posterior samples of $p(\boldsymbol{\Theta}_{0:T}, \boldsymbol{\Sigma} | \boldsymbol{Y}_{1:T})$ from Algorithm 5, functions $\tilde{\boldsymbol{F}}_{1:T}$, $\boldsymbol{J}_{1:T}$, $\boldsymbol{g}$, hyperparameters n_0, d_0, b, τ_0^2, $\boldsymbol{u}_0$, $\boldsymbol{\rho}^{(0)}$, $\boldsymbol{\eta}^{(0)}$, $\boldsymbol{m}_0^z$, $\boldsymbol{M}_0^z$, $\boldsymbol{U}$, $\boldsymbol{\Upsilon}$.
Initialize: $\boldsymbol{\phi}^{(0)} = \boldsymbol{g}(\boldsymbol{\rho}^{(0)}, \boldsymbol{\eta}^{(0)})$, $\tau_{1:T}^{2,(0)}$, $\boldsymbol{u}_{1:T}^{(0)}$, $\boldsymbol{y}_{1:T}^{(0)}$.
for $l = 1$ to L **do**
  $\tau_{1:T}^{2,(l)} \leftarrow$ `TauSq_post`$\left(\boldsymbol{u}_{0:T}^{(l-1)}, \boldsymbol{y}_{1:T}^{(l-1)}, \boldsymbol{z}_{1:T}, n_0, d_0, b, \boldsymbol{U}\right)$ ▷ Algorithm 6
  $\boldsymbol{u}_{1:T}^{(l)} \leftarrow$ `FFBS_calibration`$\left(\tau_{0:T}^{2,(l)}, \boldsymbol{y}_{1:T}^{(l-1)}, \boldsymbol{z}_{1:T}, \boldsymbol{m}_0^z, \boldsymbol{M}_0^z, \boldsymbol{U}\right)$ ▷ Algorithm 8
  $\boldsymbol{\phi}^{(l)} \leftarrow$ `Metrop`$(\boldsymbol{\phi}^{(l-1)}, \texttt{loglik}, \boldsymbol{\Upsilon})$ ▷ Algorithm 9,10
  $\boldsymbol{y}_{1:T}^{(l)} \leftarrow$ `y_post`$\left(\boldsymbol{\eta}^{(l)}, \tau_{1:T}^{2,(l)}, \boldsymbol{u}_{1:T}^{(l)}, \boldsymbol{z}_{1:T}, \tilde{\mathcal{S}}\right)$ ▷ Algorithm 7
end for ▷ $O(LT\tilde{\mathcal{S}}^3)$

$(\log(\eta_i) - \mu_i)/\sigma_i$ is the function that takes in $\boldsymbol{\eta}$ and outputs a 4-dimensional vector into $\mathbb{R}^4$ that is distributed $\mathcal{N}_4(0, I)$, and its Jacobian takes the simple form $\mathcal{J}(\boldsymbol{\eta}) = \prod_{i=1}^4 (\eta_i \sigma_i)^{-1}$.

For our noisy field observations, we utilize the data recorded on the Canadian lynx and snowshoe hare population sizes. (Hewitt, 1917) We take our training data and emulation parameters from Section 5.1 for calibration, with $N = 50$ lognormal $\boldsymbol{\eta}$ sampled with a latin square design and $T = 20$. Figure 9 plots the simulated prey and predator log-populations with the ground-truth log-populations of the hares and lynxes respectively. The agreement between the collection of curves and the real data is remarkable especially given that only the initial population was used from the real data to generate the curves and the Lotka-Volterra parameters were independently generated.

Figure 10 displays calibration samples on real lynx and hare population data collected over a 20-year period. The samples are generated using Algorithm 11 with $L = 20,000$. The resulting medians generated from the quantiles are: 0.308 $(0.036, 1.224)$, 0.054 $(0.008, 0.340)$, 0.305 $(0.09, 2.189)$, and 0.039 $(0.012, 0.097)$. Taking the medians as the point estimates for the calibrated solutions of the real-life lynx and hare population data, the number of hares grows at a rate of 0.308 times its population that year every year independent of the presence of any predators, but decreases at the rate of 0.054 times the populations of the hares and lynxes per year due to predation from the lynxes. At the same time, the number of lynxes decreases at the rate of 0.305 times its population that year, but grows at the rate of 0.039 times the populations of the hares and lynxes that year, due to eating the hares.

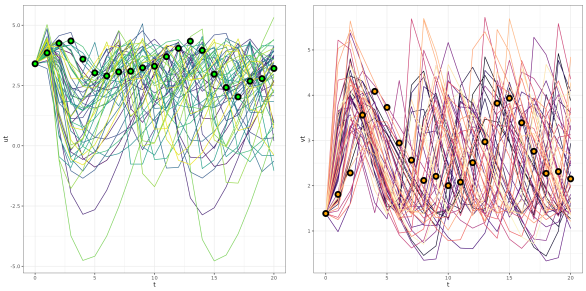

Figure 9: The simulated trajectories for 50 different parameter values of the Lotka-Volterra equations (colored splines) along with the ground-truth log-populations of hares (left) and lynxes (right). The populations of the hares and lynxes are expressed in the points in each plot.

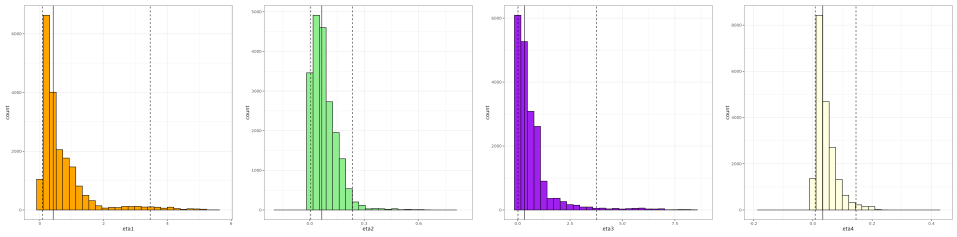

Figure 10: 20,000 samples of $\boldsymbol{\eta}$ drawn with the calibrator on real Canadian lynx and snowshoe hare population data. The black vertical line denotes the median of the samples, while the dotted lines denote the 2.5th and 97.5th percentiles.

7.2 Nonlinear Partial Differential Equation Calibration

Applying calibration to the SIR model equation (26), we fix $\boldsymbol{\rho}$ to be constant (approximately 0.85) and set the prior of $\boldsymbol{\eta}$ to be a multivariate uniform distribution $\prod_{i=1}^{5} U(a_i, b_i)$, where a_i and b_i are the limits of the uniform distribution for each parameter. Then $g_i(\eta_i) = \text{logit}((\eta_i - a_i)/(b_i - a_i))$ and the Jacobian takes on the form: $\mathcal{J}(\boldsymbol{\phi}) = \prod_{i=1}^{l_{\boldsymbol{\phi}}} \left\{(\eta_i - a_i)^{-1} + (b_i - \eta_i)^{-1}\right\}$.

In this example, we consider a 6×6 grid with 36 locations and 26 time points, including time 0, and 50 inputs $\boldsymbol{\eta}$ from a deterministic system. Carrying over our analysis from emulation in Section 5, we adopt the same AR2 structure for $\boldsymbol{F}_t$. We use 30,000 samples, with the transfer learning parameters specified in Section 4 set as $r = 50$ and $c = 2$. In calibration, we set $\alpha_1 = \alpha_3 = 0$ to simplify the analysis.

Figure 11 shows the histograms of posterior densities for elements of parameters $\boldsymbol{\eta}$ indicating that 95% credible intervals cover the true value and shrink our belief of distribution of $\boldsymbol{\eta}$ to a narrower interval. The posterior medians (95% credible intervals) of $\boldsymbol{\eta}$'s are 2.91 (2.40, 3.35) for η_1, 0.30 (0.24, 0.37) for η_2, and 0.11 (0.07, 0.20) for α_2, respectively. Figure 12 compares the emulation field generated by true and 95% credible intervals of $\boldsymbol{\eta}$. The 2.5% percentile of posterior values of $\boldsymbol{\eta}$ shows a slower process in infection and recovery, while 97.5% percentile shows a faster rate in infection and recovery over space. Figure 13 shows the

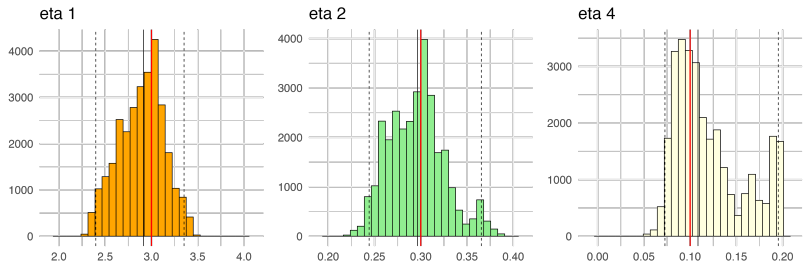

Figure 11: Histograms of posterior densities for elements of parameters $\boldsymbol{\eta}$. The red solid line represents the true value, the black solid line marks the median of the MCMC samples, and the black dotted lines indicate 95% credible intervals.

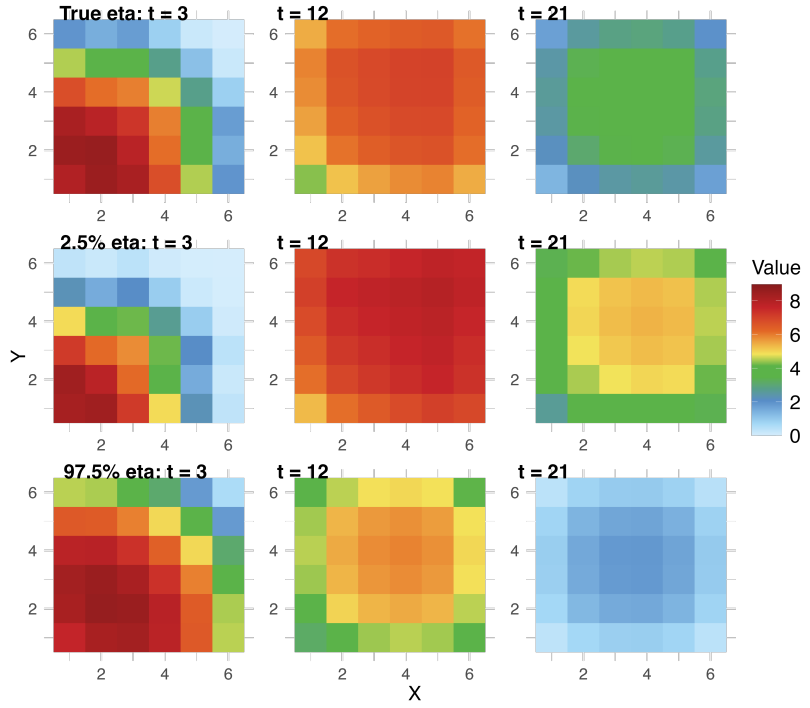

Figure 12: Heat maps of the spatial field generated by the PDE at $t = 3, 12, 21$. The first row is generated from the true $\boldsymbol{\eta}$, while the second and third rows are generated from the 2.5% and 97.5% quantiles of posterior samples of $\boldsymbol{\eta}$.

posterior variability, spatial bias, and field predictions in calibration models. The discrepancy in τ^2 with the true value arises from using only a single realization. However, both the spatial bias and field values show good alignment with the true data.

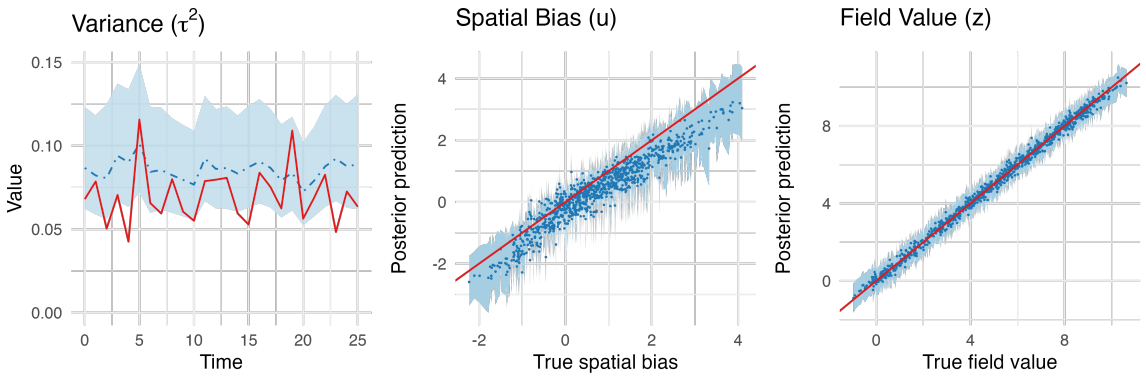

Figure 13: Joint visualization of posterior variability, spatial bias, and field predictions in calibration models, showing the median (blue dotted line and scatter points) with 95% credible intervals (light blue bands). Red lines indicate a perfect match.

7.3 Calibrating a Computer Simulation with Network Output

Network science studies how the macro-structure 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, but less attention has been given to calibrating these models to real-world data (Vega Yon and Valente, 2021; Siew, 2019). In this example, 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 a network diffusion simulation but ignores the calibration problem, justifying use of our methodology.

We first 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:

- $\text{reservoir}(t, n) = r \times \text{inflow}(t, n)$.
- $\text{outflow}(t, n) = \dfrac{(1-d)(1-r) \times \text{inflow}(t, n)}{\deg(n)}$.

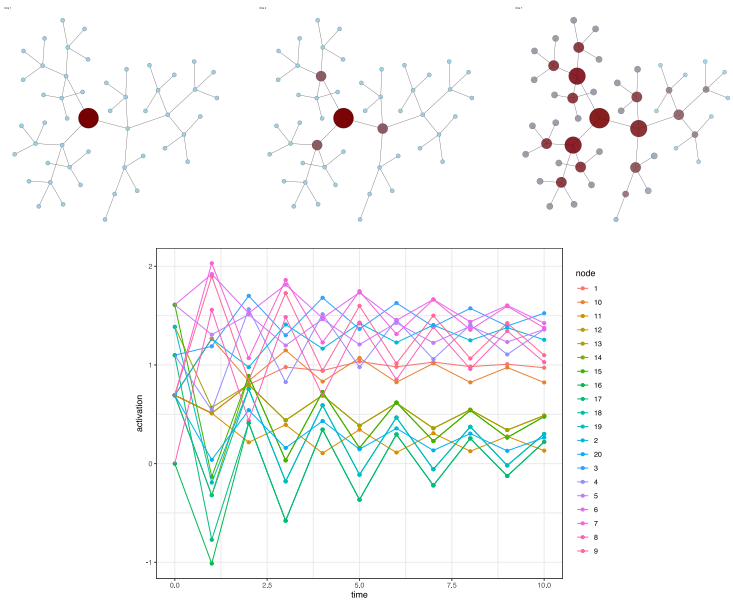

Figure 14: Individual node dynamics across the network. Each line corresponds to an activation level for a specific node over time.

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

The computer simulation then computes these quantities for each node in the network across time. A visualization of the spatiotemporal spreading dynamics is show in Figure 14.

We employ a space-filling design to generate a small collection of plausible parameter values. Subsequently, we select a random validation point that is not included in the training set. After 10,000 posterior draws, the prior-to-posterior learning is evident in Figure 15 and demonstrates our methodology is capable of calibrating a computer model generating dynamics across a network. Although it is possible to define a Gaussian process over a graph (Venkitaraman et al., 2020), for this application we fix the correlation structure to arise from the adjacency matrix. Our results demonstrate that failing to account for spatial structure leads to information loss. Specifically, we observe a 13.7% improvement in RMSE, decreasing from 0.110 to 0.095, when using the spatial model compared to the heterogeneous model, based on 25 computer model runs.

8. Concluding remarks

This manuscript demonstrates how statistical distributions can be used to devise a viable and effective probabilistic learning framework for emulating dynamically evolving spatiotemporal mechanistic systems without resorting to iterative inferential algorithms. Subsequently, we offer modularized calibration from field observations by regressing them on the emulated field. While this step requires MCMC for full probabilistic inference on the mechanistic model parameters, we do not need to recompute or update the emulated field. Building upon a rich

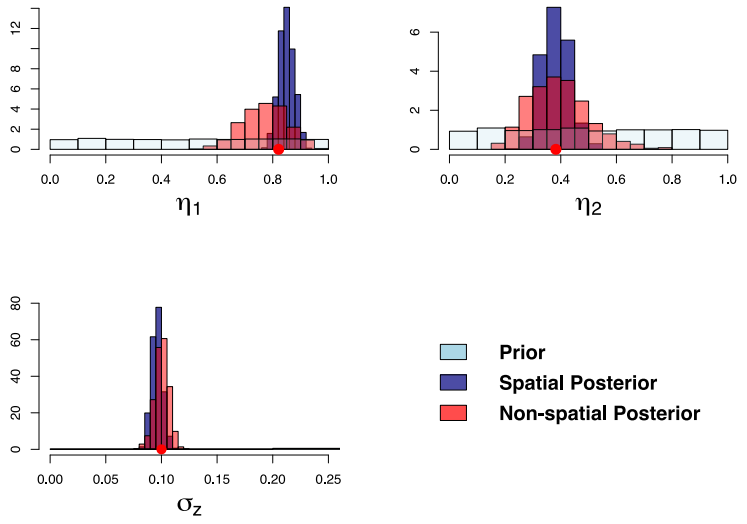

Figure 15: Network diffusion calibration posteriors. The lower uncertainty in the hierarchical model indicates pooling of information across node locations, lost in the heterogeneous model. Posterior mode point estimates are roughly similar.

literature on state-space extensions to nonlinear systems (such as nonlinear PDEs explored in Wang et al., 2021, but without melding of field observations), our approach can learn about mechanistic model parameters using more accurate and flexible emulators. Since we do not require accessibility to the mechanistic system (only its output is needed), this framework is equally applicable to systems built on mathematical models from physical principles and to agent-based models driven by computer programs (e.g., our network example).

A key point of our framework is that we emulate the mechanistic system using analytically accessible matrix-variate statistical distributions without resorting to iterative algorithms that may consume resources until they converge. Furthermore, scaling to large number of locations and time points is also achieved in closed form using Bayesian transfer learning with the FFBS algorithm. Full uncertainty quantification for learning the mechanistic system's parameters will involve MCMC, but modularization again helps with computational feasibility. In this regard, our framework holds promise as an effective trainer for generative Artificially Intelligent engines such as "BayesFlow" (Radev et al., 2022) or neural Bayes (Sainsbury-Dale et al., 2024; Zammit-Mangion et al., 2024). Such developments pertaining to the use of our framework for amortized inference will constitute future research.

In the interest of retaining exact inference, we fixed the covariance kernel hyper-parameters in the Gaussian processes used to emulate the system. This does not affect the emulated fields since, as we have remarked, the Gaussian process interpolates the field irrespective of the values of the hyper-parameters. Fixing these parameters can be achieved using exploratory methods for stochastic processes such as variograms (Zhang et al., 2019). Alternatively, we can use predictive stacking to arrive at posterior distributions averaged

over fixed values of the hyper-parameters (Zhang et al., 2024; Presicce and Banerjee, 2024). These approaches will still offer emulation without resorting to iterative MCMC algorithms. Alternatively, sequential screening procedure may be employed to identify the most influential kernel parameters (Welch et al., 1992; Merrill et al., 2021).

Learning from massive amounts of high-resolution spatial-temporal field data has not been explicitly addressed here, but the rapidly evolving literature on scaling Gaussian processes to massive data is relevant. While traditional low-rank or inducing point methods such as Snelson and Ghahramani (2005), Wilson and Nickisch (2015) or the Gaussian predictive process (Banerjee et al., 2008) remain feasible either over the set of spatial locations or over the space of inputs, it is well established that such methods can over-smooth when the number of outputs are very large (Banerjee, 2017) and multi-resolution adaptations (Nychka et al., 2015; Katzfuss, 2017) are needed to mitigate over-smoothing. Sparsity-inducing processes such as the Nearest-Neighbor Gaussian processes (Datta et al., 2016a,b; Finley et al., 2019; Peruzzi et al., 2022) or the similarly themed Vecchia-based processes (Katzfuss and Guinness, 2021; Sauer et al., 2023b,a; Cao et al., 2023) offer greater scalability and can be used to model $\boldsymbol{u}_t(\cdot)$ in Section 6. Alternatively, we can extend the divide and conquer paradigm used for emulation (Section 4) to calibration. However, the transfer learning framework may be hampered by irregular space-time coordinates. Future directions will explore spatial meta-kriging (Guhaniyogi and Banerjee, 2018; Guhaniyogi et al., 2017) and predictive stacking (Presicce and Banerjee, 2024; Pan et al., 2024).

Other extensions include treating outputs on different coordinates when adaptive grids are used to solve PDEs yielding misaligned outputs, i.e., not all runs are made over the same set of locations. Here, one may need to vectorize the incomplete output matrices and use multivariate normal distributions that would render learning using process-based misalignment models (see, e.g., Chapter 7 in Banerjee et al., 2014). High-dimensional spatial outputs can be treated using dimension-reduction methods such as spatial factor models (Ren and Banerjee, 2013; Zhang and Banerjee, 2022). One could also generalize the model to have different variances at different spatial grids. In fact, richer inference is possible using MCMC algorithms that we have avoided (except to execute modularized calibration for the mechanistic parameters). The `RobustGaSP` package developed in Gu et al. (2019) is a viable candidate to achieve effective inference in these settings. Extensions to generalized dynamic linear models (West et al., 1985; Gamerman et al., 2013) will include analyzing non-Gaussian data and investigating richer specifications for $\boldsymbol{G}_t$ and $\boldsymbol{W}_t$. We can adapt our approach using vectorized multivariate Gaussian distributions to emulate mechanistic systems at unseen spatial locations and estimate (calibrate) in spatially misaligned settings, where the emulation locations and field measurements do not match (Banerjee et al., 2014). Lastly, the burgeoning field of sequential Monte Carlo analysis is applicable to build more general statistical emulators within this framework (Hirt and Dellaportas, 2019).

9. Computing environments and timings

We developed our methods in `C++` with functions available in `R` (version 4.4.1) employing the Rcpp package (Eddelbuettel and François, 2011). Computations for the predator-prey system were executed on a laptop running 64-bit Windows 11 with a 12th Gen Intel(R) Core(TM) i7-12700H, 2.30 GHz processor, 32.0 GB RAM at 4800 MHz, 6 GB Graphics Card

equipped with multiple GPUs. Computations pertaining to the PDE and network diffusion examples were executed on a laptop running macOS 14.7.6, equipped with an Apple M3 Max processor (16-core CPU, 40-core GPU) and 64 GB of RAM. All computer programs required to reproduce the numerical results in this manuscript are available from the GitHub repository https://github.com/xiangchen-stat/Bayesian_Modeling_Mechanistic_Systems.

Empirical computation time required to solve and emulate the Lotka-Volterra ODE system, as described in Section 5.1, was approximately 0.26 and 2.16 seconds, respectively. Computations described in Section 5.2 for the PDE system required approximately 9 minutes and 13 seconds for solving the system, and 21, 61, and 59 seconds to emulate the field for the three considered variance structures, respectively. Calibration experiments, as described in Section 7, delivered full posterior inference in 98 minutes for the predator-prey system, 59 minutes for the PDE system, and about 14 and 25 seconds for the non-spatial and spatial models in the network diffusion system, respectively.

Acknowledgments

Sudipto Banerjee acknowledges funding from the National Science Foundation through DMS-1916349 and DMS-2113778; and funding from the National Institute of Health through NIEHS-R01ES027027, NIEHS-R01ES030210 and NIGMS-R01GM148761. The work of all the authors were supported, in part, from these funds.

10. Appendix

10.1 Distributions in the FFBS Algorithms 3, 4 and 5

The joint posterior distribution (4) is obtained by exploiting familiar distribution theory from statistical linear models. The first two equations in (2) implies a well-defined joint distribution for $\boldsymbol{Y}_t$ and $\boldsymbol{\Theta}_t$ conditional on $\boldsymbol{Y}_{1:(t-1)}$ as

$$\begin{bmatrix}\text{vec}(\boldsymbol{Y}_t)\\ \text{vec}(\boldsymbol{\Theta}_t)\end{bmatrix} \;\Big|\; \boldsymbol{Y}_{1:(t-1)}, \boldsymbol{\Sigma} \sim \mathcal{N}_{(N+p)S\times 1}\left(\begin{bmatrix}\text{vec}(\boldsymbol{q}_t)\\ \text{vec}(\boldsymbol{a}_t)\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}\otimes\boldsymbol{Q}_t & \boldsymbol{\Sigma}\otimes(\boldsymbol{F}_t\boldsymbol{A}_t)\\ \boldsymbol{\Sigma}\otimes\left(\boldsymbol{A}_t\boldsymbol{F}_t^\top\right) & \boldsymbol{\Sigma}\otimes\boldsymbol{A}_t\end{bmatrix}\right) . \quad (30)$$

Using familiar properties of the multivariate normal distribution, we obtain

$$\text{vec}(\boldsymbol{\Theta}_t) \mid \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma} \sim \mathcal{N}_{pS\times 1}\left(\text{vec}(\boldsymbol{a}_t) + \left(\boldsymbol{I}_S \otimes \left(\boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\right)\right)\text{vec}(\boldsymbol{Y}_t - \boldsymbol{q}_t), \boldsymbol{\Sigma}\otimes\boldsymbol{M}_t\right) , \quad (31)$$

which implies $\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_t, \boldsymbol{\Sigma} \sim \mathcal{MN}(\boldsymbol{m}_t, \boldsymbol{M}_t, \boldsymbol{\Sigma})$, where $\boldsymbol{m}_t = \boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{q}_t)$ and $\boldsymbol{M}_t = \boldsymbol{A}_t - \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\boldsymbol{F}_t\boldsymbol{A}_t$. The posterior distribution of $\boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:t}$ is obtained from

$$\begin{aligned} p(\boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:t}) &= \underbrace{\mathcal{IW}(\boldsymbol{\Sigma} \mid n_{t-1}, \boldsymbol{D}_{t-1})}_{p(\boldsymbol{\Sigma}\mid\boldsymbol{Y}_{1:(t-1)})} \times \underbrace{\mathcal{MN}(\boldsymbol{Y}_t \mid \boldsymbol{q}_t, \boldsymbol{Q}_t, \boldsymbol{\Sigma})}_{p(\boldsymbol{Y}_t\mid\boldsymbol{Y}_{1:(t-1)},\boldsymbol{\Sigma})} \\ &\propto \frac{\exp\left(-\frac{1}{2}\text{tr}\left[\boldsymbol{D}_{t-1}\boldsymbol{\Sigma}^{-1}\right]\right)}{(\det(\boldsymbol{\Sigma}))^{\frac{n_{t-1}+S+1}{2}}} \times \frac{\exp\left(-\frac{1}{2}\text{tr}\left[(\boldsymbol{Y}_t - \boldsymbol{q}_t)^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{q}_t)\boldsymbol{\Sigma}^{-1}\right]\right)}{(\det(\boldsymbol{\Sigma}))^{\frac{N}{2}}} \quad (32) \\ &\propto \frac{\exp\left(-\frac{1}{2}\text{tr}\left[\boldsymbol{D}_t\boldsymbol{\Sigma}^{-1}\right]\right)}{\det(\boldsymbol{\Sigma})^{\frac{n_t+S+1}{2}}} \propto \mathcal{IW}(\boldsymbol{\Sigma} \mid n_t, \boldsymbol{D}_t) , \end{aligned}$$

where $n_t = n_{t-1} + N$ and $\boldsymbol{D}_t = \boldsymbol{D}_{t-1} + (\boldsymbol{Y}_t - \boldsymbol{q}_t)^\top \boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t - \boldsymbol{q}_t)$. Forward sampling computes the above quantities recursively to arrive at (31) and (32) for $t = T$ and draws L samples of $\boldsymbol{\Sigma}$ from (32) followed by one draw of $\boldsymbol{\Theta}_T$ from (31) for each of the L values of $\boldsymbol{\Sigma}$.

The forward sampling described above yields L samples of $\boldsymbol{\Theta}_T, \boldsymbol{\Sigma}$ from (4). For each of the L values, we will repeat an entire cycle of backward sampling. From (7), backward sampling will draw from $p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Sigma}, \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t})$, which can be achieved by drawing one value of $\boldsymbol{\Theta}_t$ for each of the L sampled values of $(\boldsymbol{\Theta}_{t+1}, \boldsymbol{\Sigma})$ sequentially from $t = T-1, \ldots, 0$. Alternatively, we can draw the samples from $p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Sigma}, \boldsymbol{Y}_{1:t})$ directly as described in (16). To derive this distribution, we make use of the Markovian structure in the model to note that

$$p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t+1:T}, \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma}) \propto p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma}) \times p(\boldsymbol{\Theta}_{t+1} \mid \boldsymbol{\Theta}_t, \boldsymbol{\Sigma}) \propto p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma}) \,,$$

where we have used the facts that $\boldsymbol{\Theta}_t$ is conditionally independent of $\boldsymbol{Y}_{t+1:T}$ given $\boldsymbol{Y}_{1:t}$ and that $\boldsymbol{\Theta}_{t+1}$ is conditionally independent of $\boldsymbol{Y}_{1:T}$ given $\boldsymbol{\Theta}_t$. The joint distribution of $\boldsymbol{\Theta}_t$ and $\boldsymbol{\Theta}_{t+1}$ conditional on $\boldsymbol{Y}_{1:t}$ is given by

$$\begin{bmatrix} \mathrm{vec}(\boldsymbol{\Theta}_{t+1}) \\ \mathrm{vec}(\boldsymbol{\Theta}_t) \end{bmatrix} \,\Big|\, \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma} \sim \mathcal{N}_{2pS\times 1}\left(\begin{bmatrix} \mathrm{vec}(\boldsymbol{a}_{t+1}) \\ \mathrm{vec}(\boldsymbol{m}_t) \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}\otimes \boldsymbol{A}_{t+1} & \boldsymbol{\Sigma}\otimes(\boldsymbol{G}_{t+1}\boldsymbol{M}_t) \\ \boldsymbol{\Sigma}\otimes\left(\boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\right) & \boldsymbol{\Sigma}\otimes \boldsymbol{M}_t \end{bmatrix} \right) .$$

Therefore, the conditional distribution of $\boldsymbol{\Theta}_t$ given $\boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma}$ is

$$\mathrm{vec}(\boldsymbol{\Theta}_t) \mid \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma} \sim \mathcal{N}_{pS\times 1}\Big(\mathrm{vec}(\boldsymbol{m}_t) + \left(\boldsymbol{I}_S \otimes \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\right)\mathrm{vec}(\boldsymbol{\Theta}_{t+1} - \boldsymbol{a}_{t+1}), \\ \boldsymbol{\Sigma}\otimes\left(\boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t\right)\Big) \,, \quad (33)$$

which implies that $p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t+1}\boldsymbol{Y}_{1:t}, \boldsymbol{\Sigma})$ is equal to

$$\mathcal{MN}\left(\boldsymbol{\Theta}_t \mid \boldsymbol{m}_t + \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\left(\boldsymbol{\Theta}_{t+1} - \boldsymbol{a}_{t+1}\right), \boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t, \boldsymbol{\Sigma}\right) \,.$$

This also equals $p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t+1}, \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma})$ because $\boldsymbol{\Theta}_t$ and $\boldsymbol{Y}_{t+1:T}$ are conditionally independent given $\boldsymbol{\Theta}_{t+1}$. Denoting $\boldsymbol{\Theta}_{t+1} \mid \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma} \sim \mathcal{MN}\left(\boldsymbol{h}_{t+1}, \boldsymbol{H}_{t+1}, \boldsymbol{\Sigma}\right)$, where $\boldsymbol{h}_T = \boldsymbol{m}_T$ and $\boldsymbol{H}_T = \boldsymbol{M}_T$, we derive the joint distribution $p(\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_{t+1} \mid \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma}) = p(\boldsymbol{\Theta}_{t+1} \mid \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma}) \times p(\boldsymbol{\Theta}_t \mid \boldsymbol{\Theta}_{t+1}\boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma})$, which is a composition of two matrix-variate linear models with independent error matrices constructed over the same probability space (conditional on $\boldsymbol{Y}_{1:T}$ and $\boldsymbol{\Sigma}$),

$$\boldsymbol{\Theta}_t = \boldsymbol{m}_t + \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\left(\boldsymbol{\Theta}_{t+1} - \boldsymbol{a}_{t+1}\right) + \boldsymbol{E}_t \quad \text{and} \quad \boldsymbol{\Theta}_{t+1} = \boldsymbol{h}_{t+1} + \tilde{\boldsymbol{E}}_{t+1}, \qquad (34)$$

where $\boldsymbol{E}_t \sim \mathcal{MN}\left(\boldsymbol{O}, \boldsymbol{M}_t - \boldsymbol{M}_t\boldsymbol{G}_{t+1}^\top\boldsymbol{A}_{t+1}^{-1}\boldsymbol{G}_{t+1}\boldsymbol{M}_t, \boldsymbol{\Sigma}\right)$ is distributed independently of $\tilde{\boldsymbol{E}}_{t+1} \sim \mathcal{MN}\left(\boldsymbol{O}, \boldsymbol{H}_{t+1}, \boldsymbol{\Sigma}\right)$. Substituting the model for $\boldsymbol{\Theta}_{t+1}$ into the first equation in (34) followed by familiar multivariate normal calculations yields $p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T}, \boldsymbol{\Sigma}) = \mathcal{MN}\left(\boldsymbol{h}_t, \boldsymbol{H}_t, \boldsymbol{\Sigma}\right)$, where $\boldsymbol{h}_t$ and $\boldsymbol{H}_t$ are defined recursively as in (16). This completes the required closed-form derivations for all the distributions in the exact FFBS algorithm.

10.2 Calculus-free derivation of $\mathcal{HT}$ density

Recall that in equations (5) and (6) we were able to compute their distributions by means of marginalizing, that is, integrating, out the undesired variance terms. We demonstrate an approach to this problem without using integration. Begin with the identity

$$p(A \mid B)p(B) = p(A, B) \qquad (35)$$

where A and B denote random variables. Let $p(A \mid B) = p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T})$, $p(B) = p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T})$, and $p(A, B) = p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$. Dividing both sides of (35) by $p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T})$, we get:

$$p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T}) = \frac{p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})}{p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T})} \tag{36}$$

Recall that $p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$ is matrix-normal-inverse-Wishart with density (follows from (3):)

$$p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma}_T \mid \boldsymbol{Y}_{1:T}) = \frac{(\det(\boldsymbol{D}_T))^{n_T/2} \exp\left(-\frac{1}{2}\mathrm{tr}\left\{\left[\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right]\boldsymbol{\Sigma}^{-1}\right\}\right)}{2^{n_T S/2}\,(2\pi)^{pS/2}\,\Gamma_S(\frac{n_T}{2})\,(\det(\boldsymbol{H}_t))^{S/2}\,(\det(\boldsymbol{\Sigma}))^{\frac{n_T+S+p+1}{2}}}\,, \tag{37}$$

We are also given that $p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T})$ is an inverse-Wishart. More specifically, $p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T}) \propto p(\boldsymbol{\Theta}_t, \boldsymbol{\Sigma} \mid \boldsymbol{Y}_{1:T})$, which means they share product terms in their respective expressions that contain $\boldsymbol{\Sigma}$ and $\boldsymbol{\Theta}_t$ (and $\boldsymbol{Y}_{1:T}$). Hence:

$$p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T}) \propto \frac{\exp\left(-\frac{1}{2}\mathrm{tr}\left\{\left[\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right]\boldsymbol{\Sigma}^{-1}\right\}\right)}{(\det(\boldsymbol{\Sigma}))^{\frac{n_T+S+p+1}{2}}} \tag{38}$$

Introducing the constant (non-$\boldsymbol{\Sigma}$) multiplier for (38), given it is Inverse-Wishart, yields:

$$\begin{aligned} p(\boldsymbol{\Sigma} \mid \boldsymbol{\Theta}_t, \boldsymbol{Y}_{1:T}) =& \frac{\exp\left(-\frac{1}{2}\mathrm{tr}\left\{\left[\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right]\boldsymbol{\Sigma}^{-1}\right\}\right)}{2^{(n_T+p)S/2}\Gamma_S\left(\frac{n_T+p}{2}\right)(\det(\boldsymbol{\Sigma}))^{\frac{n_T+S+p+1}{2}}} \\ & \times \left(\det\left(\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right)\right)^{(n_T+p)/2} \end{aligned} \tag{39}$$

Finally, we take the quotient to obtain the exact form of $p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T})$:

$$\begin{aligned} p(\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_{1:T}) =& \frac{(\det(\boldsymbol{D}_T))^{n_T/2} \exp\left(-\frac{1}{2}\mathrm{tr}\left\{\left[\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right]\boldsymbol{\Sigma}^{-1}\right\}\right)}{2^{n_T S/2}\,(2\pi)^{pS/2}\,\Gamma_S(\frac{n_T}{2})\,(\det(\boldsymbol{H}_t))^{S/2}\,(\det(\boldsymbol{\Sigma}))^{\frac{n_T+S+p+1}{2}}} \\ & \times \frac{2^{(n_T+p)S/2}\Gamma_S\left(\frac{n_T+p}{2}\right)(\det(\boldsymbol{\Sigma}))^{\frac{n_T+S+p+1}{2}}}{\exp\left(-\frac{1}{2}\mathrm{tr}\left\{\left[\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right]\boldsymbol{\Sigma}^{-1}\right\}\right)} \\ & \times \left(\det\left(\boldsymbol{D}_T + (\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right)\right)^{-(n_T+p)/2} \\ =& \pi^{-pS/2}\frac{\Gamma_S\left(\frac{n_T+p}{2}\right)}{\Gamma_S\left(\frac{n_T}{2}\right)}(\det(\boldsymbol{D}_T))^{-p/2}\,(\det(\boldsymbol{H}_t))^{S/2} \\ & \times \left(\det\left(\boldsymbol{I}_S + \boldsymbol{D}_T^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)^\top \boldsymbol{H}_t^{-1}(\boldsymbol{\Theta}_t - \boldsymbol{h}_t)\right)\right)^{-(n_T+p)/2} \end{aligned} \tag{40}$$

This is the Hyper-T density in (5), specifically $\mathcal{HT}(\boldsymbol{\Theta}_t \mid \boldsymbol{h}_t, \boldsymbol{H}_t, n_T, \boldsymbol{D}_T)$. We have thus demonstrated a derivation of the Hyper-T density without using integration.

10.3 Distributions in the FFBS algorithm for the model in (14)

The joint posterior distribution corresponding to the model in (14) is obtained by modifying (30). The first two equations in (14) implies the joint distribution for $\boldsymbol{Y}_t$ and $\boldsymbol{\Theta}_t$ conditional on $\boldsymbol{Y}_{1:(t-1)}$ as

$$\begin{bmatrix}\text{vec}(\boldsymbol{Y}_t)\\ \text{vec}(\boldsymbol{\Theta}_t)\end{bmatrix} \;\Big|\; \boldsymbol{Y}_{1:(t-1)},\, \sigma^2 \sim \mathcal{N}_{(N+p)S\times 1}\left(\begin{bmatrix}\text{vec}(\boldsymbol{q}_t)\\ \text{vec}(\boldsymbol{a}_t)\end{bmatrix}, \sigma^2\begin{bmatrix}\boldsymbol{R}\otimes\boldsymbol{Q}_t & \boldsymbol{R}\otimes(\boldsymbol{F}_t\boldsymbol{A}_t)\\ \boldsymbol{R}\otimes\left(\boldsymbol{A}_t\boldsymbol{F}_t^\top\right) & \boldsymbol{R}\otimes\boldsymbol{A}_t\end{bmatrix}\right). \tag{41}$$

We obtain the analog of (31)

$$\text{vec}(\boldsymbol{\Theta}_t) \mid \boldsymbol{Y}_{1:t},\, \sigma^2 \sim \mathcal{N}_{pS\times 1}\left(\text{vec}(\boldsymbol{a}_t) + \left(\boldsymbol{I}_S\otimes\left(\boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\right)\right)\text{vec}(\boldsymbol{Y}_t-\boldsymbol{q}_t), \sigma^2\boldsymbol{R}\otimes\boldsymbol{M}_t\right), \tag{42}$$

which implies $\boldsymbol{\Theta}_t \mid \boldsymbol{Y}_t, \sigma^2 \sim \mathcal{MN}\left(\boldsymbol{m}_t, \boldsymbol{M}_t, \sigma^2\boldsymbol{R}\right)$, where $\boldsymbol{m}_t = \boldsymbol{a}_t + \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t-\boldsymbol{q}_t)$ and $\boldsymbol{M}_t = \boldsymbol{A}_t - \boldsymbol{A}_t\boldsymbol{F}_t^\top\boldsymbol{Q}_t^{-1}\boldsymbol{F}_t\boldsymbol{A}_t$. The posterior distribution of $\sigma^2 \mid \boldsymbol{Y}_{1:t}$ is obtained as

$$\begin{aligned}
p(\sigma^2 \mid \boldsymbol{Y}_{1:t}) &= \underbrace{\mathcal{IG}\left(\sigma^2 \mid n_{t-1}, d_{t-1}\right)}_{p(\sigma^2\mid\boldsymbol{Y}_{1:(t-1)})} \times \underbrace{\mathcal{MN}(\boldsymbol{Y}_t \mid \boldsymbol{q}_t, \boldsymbol{Q}_t, \sigma^2\boldsymbol{R})}_{p(\boldsymbol{Y}_t\mid\boldsymbol{Y}_{1:(t-1)},\sigma^2)}\\
&\propto \frac{1}{\sigma^{2(n_{t-1}+1)}} \times \exp\left(-\frac{d_{t-1}}{\sigma^2}\right) \times \frac{\exp\left(-\frac{1}{2\sigma^2}\text{tr}\left[(\boldsymbol{Y}_t-\boldsymbol{q}_t)^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t-\boldsymbol{q}_t)\boldsymbol{R}^{-1}\right]\right)}{\sigma^{2(\frac{NS}{2})}}\\
&\propto \frac{1}{\sigma^{2(n_{t-1}+\frac{NS}{2}+1)}} \times \exp\left(-\frac{1}{\sigma^2}\left(d_{t-1} + \frac{1}{2}\text{tr}\left[(\boldsymbol{Y}_t-\boldsymbol{q}_t)^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t-\boldsymbol{q}_t)\boldsymbol{R}^{-1}\right]\right)\right)\\
&\propto \mathcal{IG}\left(\sigma^2 \mid n_t, d_t\right),
\end{aligned} \tag{43}$$

where $n_t = n_{t-1} + \frac{NS}{2}$ and $d_t = d_{t-1} + \frac{1}{2}\text{tr}\left[(\boldsymbol{Y}_t-\boldsymbol{q}_t)^\top\boldsymbol{Q}_t^{-1}(\boldsymbol{Y}_t-\boldsymbol{q}_t)\boldsymbol{R}^{-1}\right]$. Forward sampling computes the above quantities recursively to arrive at (42) and (43) for $t = T$ and draws L samples of σ^2 from (43) followed by one draw of $\boldsymbol{\Theta}_T$ from (42) for each of the L values of σ^2.

References

N. Abdalla, S. Banerjee, G. Ramachandran, and S. Arnold. Bayesian state space modeling of physical processes in industrial hygiene. *Technometrics*, 62(2):147–160, 2020. doi: 10.1080/00401706.2019.1630009. URL `https://doi.org/10.1080/00401706.2019.1630009`.

S. Banerjee. High-dimensional Bayesian geostatistics. *Bayesian analysis*, 12(2):583, 2017.

S. Banerjee and A. Roy. *Linear algebra and matrix analysis for statistics*. CRC Press, Boca Raton, FL, 2014.

S. Banerjee, A. E. Gelfand, A. O. Finley, and H. Sang. Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(4):825–848, 2008.

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

M. J. Bayarri, J. O. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R. J. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *The Annals of Statistics*, 35(5):1874–1906, 2007a.

M. J. Bayarri, J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007b.

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

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

J. Cao, M. Kang, F. Jimenez, H. Sang, F. T. Schaefer, and M. Katzfuss. Variational sparse inverse cholesky approximation for latent Gaussian processes via double kullback-leibler minimization. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3559–3576. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/cao23b.html.

C. Carter and R. Kohn. On Gibbs Sampling for State Space Models. *Biometrika*, 1994.

K. Y. Chan and M. S. Vitevitch. The influence of the phonological neighborhood clustering coefficient on spoken word recognition. *Journal of Experimental Psychology: Human Perception and Performance*, 35(6):1934, 2009.

A. M. Chan-Golston, S. Banerjee, and M. S. Handcock. Bayesian inference for finite populations under spatial process settings. *Environmetrics*, 31, 2020.

S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651, 2010.

N. Cressie and G. Johannesson. Fixed rank kriging for very large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):209–226, 2008. doi: https://doi.org/10.1111/j.1467-9868.2007.00633.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2007.00633.x.

A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand. Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016a. doi: 10.1080/01621459.2015.1044091. URL https://doi.org/10.1080/01621459.2015.1044091. PMID: 29720777.

A. Datta, S. Banerjee, A. O. Finley, N. A. S. Hamm, and M. Schaap. Nonseparable dynamic nearest neighbor Gaussian process models for large spatio-temporal data with an application to particulate matter analysis. *The Annals of Applied Statistics*, 10(3):1286 – 1316, 2016b.

D. Dey, A. Datta, and S. Banerjee. Graphical Gaussian process models for highly multivariate spatial data. *Biometrika*, 109(4):993–1014, 2022. ISSN 1464-3510. doi: 10.1093/biomet/asab061. URL https://doi.org/10.1093/biomet/asab061.

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 2011.

S. Eleftheriadis, T. Nicholson, M. Deisenroth, and J. Hensman. Identification of Gaussian process state space models. *Advances in neural information processing systems*, 30, 2017.

K.-T. Fang, R. Li, and A. Sudjianto. *Design and Modeling for Computer Experiments*. Chapman and Hall/CRC, oct 2005.

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

A. O. Finley, A. Datta, B. C. Cook, D. C. Morton, H. E. Andersen, and S. Banerjee. Efficient algorithms for Bayesian Nearest Neighbor Gaussian Processes. *Journal of Computational and Graphical Statistics*, 28(2):401–414, 2019.

S. Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer New York, 2006.

D. Gamerman and H. F. Lopes. *Markov chain Monte Carlo*. Chapman & Hall/CRC Texts in Statistical Science. Chapman & Hall/CRC, Philadelphia, PA, 2 edition, May 2006.

D. Gamerman, T. R. dos Santos, and G. C. Franco. A non-Gaussian family of state-space models with exact marginal likelihood. *Journal of Time Series Analysis*, 34(6):625–645, 2013.

Y. Gel, A. E. Raftery, and T. Gneiting. Calibrated probabilistic mesoscale weather field forecasting. *Journal of the American Statistical Association*, 99(467):575–583, 2004. doi: 10.1198/016214504000000872. URL https://doi.org/10.1198/016214504000000872.

A. E. Gelfand and S. K. Ghosh. Model Choice: A Minimum Posterior Predictive Loss Approach. *Biometrika*, 85(1):1–11, June 1998.

R. B. Gramacy and D. W. Apley. Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578, 2015. doi: 10.1080/10618600.2014.914442. URL http://dx.doi.org/10.1080/10618600.2014.914442.

R. B. Gramacy and H. K. H. Lee. Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103 (483):1119–1130, 2008.

M. Gu. Robustcalibration: Robust calibration of computer models in r. *arXiv preprint arXiv:2201.01476*, 2022.

M. Gu and J. O. Berger. Parallel partial Gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, pages 1317–1347, 2016.

M. Gu and H. Li. Gaussian orthogonal latent factor processes for large incomplete matrices of correlated data. *Bayesian Analysis*, 17(4):1219–1244, 2022.

M. Gu and W. Shen. Generalized probabilistic principal component analysis of correlated data. *Journal of Machine Learning Research*, 21(13):1–41, 2020.

M. Gu, J. Palomo, and J. O. Berger. Robustgasp: Robust Gaussian stochastic process emulation in r. *R Journal*, 11(1), 2019.

R. Guhaniyogi and S. Banerjee. Meta-kriging: Scalable Bayesian modeling and inference for massive spatial datasets. *Technometrics*, 60(4):430–444, 2018.

R. Guhaniyogi, C. Li, T. D. Savitsky, and S. Srivastava. A divide-and-conquer Bayesian approach to large-scale kriging. *arXiv preprint arXiv:1712.09767*, 2017.

A. K. Gupta and D. K. Nagar. *Matrix variate distributions*. CRC Press, 1999.

R. Haylock and A. O'Hagan. *Bayesian Statistics*, volume 5. Oxford University Press, New York, 1996.

M. J. Heaton, A. Datta, A. O. Finley, R. Furrer, J. Guinness, R. Guhaniyogi, F. Gerber, R. B. Gramacy, D. M. Hammerling, M. Katzfuss, F. Lindgren, D. W. Nychka, F. Sun, and A. Zammit-Mangion. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological, and Environmental Statistics*, 24(3):398–425, 2019.

C. G. Hewitt. Conservation of wild life in canada. *Nature*, 99:246–247, 1917.

D. Higdon, M. Kennedy, J. C. Cavendish, J. A. Cafeo, and R. D. Ryne. Combining field data and computer simulations for calibration and prediction. *SIAM Journal on Scientific Computing*, 26(2):448–466, 2004.

D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482): 570–583, 2008.

M. Hirt and P. Dellaportas. Scalable Bayesian learning for state space models using variational inference with smc samplers. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 76–86. PMLR, 2019.

X. Kang and X. Deng. Design and analysis of computer experiments with quantitative and qualitative inputs: A selective review. *WIREs Data Mining and Knowledge Discovery*, 10(3):e1358, 2020. doi: https://doi.org/10.1002/widm.1358. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1358`.

M. Katzfuss. A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112:201–214, 2017. doi: 10.1080/01621459.2015.1123632. URL `http://dx.doi.org/10.1080/01621459.2015.1123632`.

M. Katzfuss and J. Guinness. A general framework for Vecchia approximations of Gaussian processes. *Statistical Science*, 36(1):124–141, 2021. doi: 10.1214/19-STS755. URL https://doi.org/10.1214/19-STS755.

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

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

M. L. Krock, W. Kleiber, D. Hammerling, and S. Becker. Modeling massive highly multivariate nonstationary spatial data with the basis graphical lasso. *Journal of Computational and Graphical Statistics*, 0(0):1–16, 2023. doi: 10.1080/10618600.2023.2174126. URL https://doi.org/10.1080/10618600.2023.2174126.

F. Liu and M. West. A Dynamic Modelling Strategy for Bayesian Computer Model Emulation. *Bayesian Analysis*, 4(2):393–412, 2009.

E. Merrill, A. Fern, X. Fern, and N. Dolatnia. An empirical study of Bayesian optimization: acquisition versus partition. *The Journal of Machine Learning Research*, 22(1):200–224, 2021.

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

D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, and S. Sain. A multiresolution Gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2):579–599, 2015.

J. E. Oakley and A. O'Hagan. Probabilistic sensitivity analysis of complex models: a Bayesian approach. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 66(3): 751–769, 2004.

A. O'Hagan. *Bayesian Statistics*, volume 4. Oxford University Press, New York, 1992.

S. Pan, L. Zhang, J. R. Bradley, and S. Banerjee. Bayesian inference for spatial-temporal non-Gaussian data using predictive stacking, 2024. URL https://arxiv.org/abs/2406.04655.

M. Peruzzi, S. Banerjee, and A. Finley. Highly Scalable Bayesian Geostatistical Modeling via Meshed Gaussian Processes on Partitioned Domains. *Journal of the American Statistical Association*, 117(538):969–982, 2022.

G. Petris, S. Petrone, and P. Campagnoli. *Dynamic Linear Models with R*. Springer New York, 2009.

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

L. Presicce and S. Banerjee. Bayesian transfer learning for artificially intelligent geospatial systems: A predictive stacking approach. *arXiv preprint arXiv:2410.09504*, 2024.

J. Quinonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Tranacttions on Neural Networks ad Learning Systems*, 33(4):1452–1466, 2022.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Q. Ren and S. Banerjee. Hierarchical factor models for large spatially misaligned data: A low-rank predictive process approach. *Biometrics*, 69(1):19–30, 2013. doi: 10.1111/j.1541-0420.2012.01832.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1541-0420.2012.01832.x.

Q. Ren, S. Banerjee, A. O. Finley, and J. S. Hodges. Variational Bayesian methods for spatial data analysis. *Computational Statistics & Data Analysis*, 55(12):3197–3217, 2011. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2011.05.021. URL https://www.sciencedirect.com/science/article/pii/S0167947311002003.

C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Texts in Statistics. Springer, New York, NY, 2 edition, Jul 2005.

H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009.

J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409 – 423, 1989. doi: 10.1214/ss/1177012413. URL https://doi.org/10.1214/ss/1177012413.

M. Sainsbury-Dale, A. Zammit-Mangion, and R. Huser. Likelihood-free parameter estimation with neural bayes estimators. *The American Statistician*, 78(1):1–14, 2024.

T. J. Santner, B. J. Williams, and W. I. Notz. *The design and analysis of computer experiments the design and analysis of computer experiments*. Springer series in statistics. Springer, New York, NY, 2 edition, Jan. 2019.

S. Särkkä and L. Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2013.

A. Sauer, A. Cooper, and R. B. Gramacy. Vecchia-approximated deep Gaussian processes for computer experiments. *Journal of Computational and Graphical Statistics*, 32(3):824–837, 2023a. doi: 10.1080/10618600.2022.2129662. URL https://doi.org/10.1080/10618600.2022.2129662.

A. Sauer, R. B. Gramacy, and D. Higdon. Active learning for deep Gaussian process surrogates. *Technometrics*, 65(1):4–18, 2023b. doi: 10.1080/00401706.2021.2008505. URL https://doi.org/10.1080/00401706.2021.2008505.

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

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.

K. Soetaert, T. Petzoldt, and R. W. Setzer. Solving di erential equations in r: Package desolve. *Journal of Statistical Software*, 33(9), 2010.

J. R. Stroud, M. L. Stein, D. J. S. Barry M. Lesht, and D. Beletsky. An ensemble kalman filter and smoother for satellite data assimilation. *Journal of the American Statistical Association*, 105(491):978–990, 2010. doi: 10.1198/jasa.2010.ap07636. URL https://doi.org/10.1198/jasa.2010.ap07636.

R. Turner, M. Deisenroth, and C. Rasmussen. State-Space Inference and Learning with Gaussian Processes. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, 2010.

G. Vega Yon and T. Valente. netdiffuser: Analysis of diffusion and contagion processes on networks. *R package version*, 1(3), 2021.

A. Vehtari, A. Gelman, and J. Gabry. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistical Computing*, 27:1413–1432, 2017.

A. Venkitaraman, S. Chatterjee, and P. Handel. Gaussian processes over graphs. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5640–5644. IEEE, 2020.

M. Vitevitch, G. Ercal, and B. Adagarla. Simulating Retrieval from a Highly Clustered Network: Implications for Spoken Word Recognition. *Frontiers in psychology*, 2011.

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

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

W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris. Screening, predicting, and computer experiments. *Technometrics*, 34(1):15–25, 1992.

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

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

C. K. Wikle. Hierarchical Bayesian models for predicting the spread of ecological processes. *Ecology*, 84(6):1382–1394, 2003.

C. K. Wikle. Low-rank representations for spatial processes. *Handbook of Spatial Statistics*, pages 107–118, 2010. Gelfand, A. E., Diggle, P., Fuentes, M. and Guttorp, P., editors, Chapman and Hall/CRC, pp. 107-118.

C. K. Wikle and L. M. Berliner. A Bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, 230(1):1–16, 2007. ISSN 0167-2789. doi: https://doi.org/10.1016/j.physd.2006.09.017. URL https://www.sciencedirect.com/science/article/pii/S016727890600354X. Data Assimilation.

C. K. Wikle and M. B. Hooten. A general science-based framework for dynamical spatio-temporal models. *TEST*, 19(3):417–451, Nov 2010. ISSN 1863-8260. doi: 10.1007/s11749-010-0209-z. URL https://doi.org/10.1007/s11749-010-0209-z.

A. Wilson and H. Nickisch. Kernel interpolation for scalable structured Gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.

A. Zammit-Mangion, M. Sainsbury-Dale, and R. Huser. Neural methods for amortized inference. *Annual Review of Statistics and Its Application*, 12, 2024.

L. Zhang and S. Banerjee. Spatial factor modeling: A Bayesian matrix-normal approach for misaligned data. *Biometrics*, 78(2):560–573, June 2022.

L. Zhang, A. Datta, and S. Banerjee. Practical Bayesian modeling and inference for massive spatial data sets on modest computing environments. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 12(3):197–209, 2019.

L. Zhang, W. Tang, and S. Banerjee. Bayesian geostatistics using predictive stacking. *arXiv:2304.12414v2*, 2024. URL https://arxiv.org/abs/2304.12414.

T. Zhang and W. Wang. Existence of traveling wave solutions for influenza model with treatment. *Journal of Mathematical Analysis and Applications*, 419(1):469–495, 2014.

S. Zhu, K. Yu, and Y. Gong. Predictive matrix-variate t models. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/061412e4a03c02f9902576ec55ebbe77-Paper.pdf.