Improving Implicit Spatiotemporal Inference by Combining Particle Filters and Neural Posterior Estimation

Shuge Ouyang

Thesis Advisor
Professor Edward Ionides

Department of Statistics, University of Michigan

Abstract: Particle filtering methods provide a general approach for inference in moderately high-dimensional, nonlinear, non-Gaussian, partially observed Markov process models, where the latent transition density is analytically intractable. However, their performance can deteriorate due to the curse of dimensionality. We propose a particle filtering method that propagates particles through intermediate time steps between observation times, while using neural posterior estimation (NPE) to approximate the predictive likelihood of future observations. Resampling is then performed using these predictive likelihood evaluations, yielding a neural-guided intermediate resampling filter. The resulting algorithm is plug-and-play and applicable to models whose latent Markov process admits an implicit and difficult-to-handle transition density. As an illustrative example, we demonstrate the method on spatiotemporal infectious disease transmission dynamics, using a measles metapopulation model over a network of connected regions.

Keywords and phrases: sequential Monte Carlo; particle filter; spatiotemporal inference; guided intermediate resampling filter; neural posterior estimation

1 Introduction

Spatiotemporal dynamical models arise whenever many interacting units evolve over time and are only partially observed. Examples include infectious disease transmission across cities, ecological metapopulations, and monitoring networks in the environmental sciences. These systems often combine local nonlinear dynamics with weak but consequential coupling between nearby units. Reliable inference in such settings supports forecasting, scientific understanding, and decision making about interventions.

A convenient statistical framework for these problems is the partially observed Markov pro-

cess model, abbreviated as a POMP model. A POMP model combines a latent Markov process with a measurement process that links latent states to observations. In many mechanistic models the latent transition density is not available in closed form, even though forward simulation is straightforward. We refer to this setting as an implicit transition. This distinction matters because many inference methods require evaluating the transition density, whereas realistic spatiotemporal simulators often provide only sample paths.

Particle filters, also called sequential Monte Carlo methods, provide a general plug and play approach to filtering and likelihood evaluation for POMP models. A particle filter propagates an ensemble of simulated particles forward in time and repeatedly reweights and resamples particles according to how well they explain the data. This mechanism is attractive because it requires only a simulator for the latent process and an evaluator for the measurement model. However particle filters can struggle in high dimensional spatiotemporal systems. As the number of units grows, the weights can become extremely uneven, most particles receive negligible weight, and resampling collapses diversity. The phenomenon is a practical form of the curse of dimensionality and it can lead to unstable likelihood estimates and prohibitive computational cost.

Guided intermediate resampling filter methods mitigate this instability by introducing additional resampling times between observation times. Intermediate resampling can correct trajectories before they drift too far from data constraints, improving stability over long horizons and across many coupled units. The effectiveness of this strategy depends on having informative resampling weights that anticipate future observations. In classical guided filters these weights are constructed from model specific guide functions, and designing a good guide can be difficult when the transition is implicit and the dynamics are nonlinear and strongly stochastic.

This thesis aims to improve practical scalability and robustness of filtering for implicit spatiotemporal POMP models by combining intermediate resampling with neural posterior estimation, abbreviated as NPE. The central idea is to train a neural model that approximates a predictive distribution for future observations given the current latent state, and to use the resulting predictive likelihood evaluations to guide intermediate resampling. This approach seeks to retain the plug and play character of particle filtering while reducing particle collapse in regimes where hand designed guides are inadequate or costly to engineer.

This work makes several contributions. First it defines a neural guided intermediate resampling filter that replaces a hand crafted guide with an NPE based predictive likelihood approximation. Second it proposes a training target that is aligned with the quantities needed to compute resampling weights at intermediate times. Third it presents an implementation that couples an R based spatiotemporal POMP workflow with a Python based neural guide while maintaining reproducibility of experiments. Fourth it evaluates the method on a spatiotemporal infectious disease example using a measles transmission model over a network of connected regions, and compares against established guided filtering baselines and alternative guiding strategies.

The remainder of the thesis is organized as follows. Chapter two introduces spatiotemporal POMP models and reviews particle filtering with an emphasis on why weight degeneracy

worsens with dimension. Chapter two also presents guided intermediate resampling and discusses two existing guide constructions that are commonly used in practice. Chapter three introduces the proposed neural guided method, including training data generation, model architecture choices, and the training objective. Chapter four describes the measles model and the experimental design used to compare methods, including accuracy, stability, and computational cost. Chapter five summarizes findings, discusses limitations, and outlines directions for future work.

1.1 Previous Literature

Particle filtering is a standard tool for inference in partially observed Markov process models, where a latent stochastic process is observed through noisy measurements. The bootstrap particle filter of Gordon et al. (1993) popularized a simulation based approach that only requires the ability to simulate the latent process forward in time and evaluate the measurement density. This plug and play perspective is particularly useful for mechanistic models in ecology and epidemiology, where the transition density is rarely available in closed form but forward simulation is natural. The POMP framework and its computational implementations have been developed in detail by Bretó et al. (2009), and by King et al. (2016), and extended to spatiotemporal settings through software such as `spatPomp` (Asfaw et al., 2024).

A large body of theory explains why particle filters work well in low dimension and why they can fail in high dimension. Classical results establish consistency and central limit theorems for Monte Carlo estimators produced by sequential importance sampling and resampling (Chopin, 2004). Concentration inequalities and stability results for Feynman Kac particle systems were developed in work by Moral (2004) , which provides a mathematical language for analyzing resampling based algorithms. At the same time, the empirical phenomenon of weight degeneracy becomes severe as the dimension grows, and this leads to filter collapse unless the number of particles grows rapidly. Bengtsson et al. (2008) and Snyder et al. (2008) provide influential analyzes of this curse of dimensionality in nonlinear filtering.

Many approaches have been proposed to improve scalability for spatiotemporal and other high dimensional systems by introducing approximations that exploit weak coupling or locality. Ensemble Kalman filter methods introduced by Evensen (2003) and developed in later work propagate an ensemble through the nonlinear dynamics and update using a Gaussian inspired rule, which can be effective when the posterior is close to Gaussian but may struggle in strongly nonlinear or discrete settings. Block particle filters, including the formulation analyzed by Rebeschini and Van Handel (2015), reduce Monte Carlo variance by resampling in spatial blocks and can avoid exponential scaling under suitable decay of dependence. Localized particle filter variants for geophysical data assimilation, such as the work of Poterjoy (2016), similarly use local likelihood contributions to stabilize weights. Bagging based approaches provide another perspective, where an ensemble of Monte Carlo filters is combined using localized information, as developed by Ionides et al. (2023).

Guided particle filtering introduces an explicit guide function designed to approximate the predictive likelihood of future observations and to perform earlier resampling when particles appear unlikely to match future data. The auxiliary particle filter of Pitt and Shephard

(1999) is a canonical one step lookahead method that uses a guide to select promising particles before propagation. Intermediate resampling generalizes this idea by inserting additional resampling times between observation times in a continuous time or finely discretized model. Park and Ionides (2020) propose the guided intermediate resampling filter, which formalizes intermediate resampling with a principled discounting schedule and practical guide constructions, including moment based guides that can be implemented in plug and play settings. This line of work is directly related to the present thesis, since the neural guide proposed here can be viewed as a data driven replacement for analytic or moment based predictive approximations.

A separate but increasingly connected literature develops neural density estimation methods for likelihood free inference in implicit models. Sequential neural posterior estimation was introduced as a practical route to learning conditional densities from simulated pairs of parameters and data, for example in Papamakarios et al. (2019), and later work refined these ideas with flexible density estimators based on normalizing flows. Survey style discussions, such as Brehmer and Cranmer (2022) emphasize that modern simulation based inference can be interpreted as learning components of the likelihood or posterior that are otherwise intractable.

There is also growing work that connects neural inference and sequential Monte Carlo more directly by learning proposals, guides, or differentiable relaxations of resampling. Variational sequential Monte Carlo methods like (Naesseth et al., 2018), and related objectives such as filtering variational objectives, provide end to end training criteria that treat SMC as a differentiable computation graph with carefully chosen gradient estimators. Other work studies differentiable approximations to resampling or optimal transport based couplings as a way to enable gradient based learning while modifying the forward pass. These approaches share the goal of using learned components to improve the performance of particle methods, but they typically target posterior inference or learning rather than likelihood evaluation in high dimensional spatiotemporal POMP models. The method developed in this thesis instead keeps the standard GIRF style likelihood evaluation structure, and introduces neural posterior estimation as a plug in guide function for intermediate resampling.

2 Background

2.1 Partially Observed Markov Process Models and SpatPOMP

A convenient mathematical framework for nonlinear dynamical systems observed with noise is the partially observed Markov process model, abbreviated as a POMP model (King et al., 2016). In discrete time, a POMP model consists of a latent Markov process

$$X_0, X_1, \ldots, X_N$$

and a corresponding sequence of observations

$$Y_1, Y_2, \ldots, Y_N.$$

The latent process is assumed to satisfy the Markov property

$$X_n \mid X_{0:n-1} \sim f_{X_n|X_{n-1}}(\cdot \mid X_{n-1}; \theta), \qquad n = 1, \ldots, N,$$

where θ denotes the model parameter vector. The measurement process is conditionally independent given the latent state, so that

$$Y_n \mid X_{0:n}, Y_{1:n-1} \sim g_n(\cdot \mid X_n; \theta), \qquad n = 1, \ldots, N.$$

Here $f_{X_n|X_{n-1}}$ is the latent transition law and g_n is the measurement density. The observed data are denoted by

$$y_{1:N} = (y_1, \ldots, y_N),$$

and the likelihood can be written as

$$p_\theta(y_{1:N}) = \prod_{n=1}^{N} p_\theta(y_n \mid y_{1:n-1}).$$

Each one-step predictive factor has the filtering representation

$$f_{Y_n|Y_{1:n-1}}(y_n \mid y_{1:n-1}; \theta) = \int g_n(y_n \mid x_n; \theta)\, f_{X_n|Y_{1:n-1}}(x_n \mid y_{1:n-1}; \theta)\, dx_n. \tag{1}$$

so likelihood evaluation is inseparable from filtering. In other words, to compute the likelihood one must repeatedly approximate the predictive distribution of the latent state and integrate the measurement model against it.

The spatiotemporal models considered in this thesis add an additional unit structure to the POMP framework. Suppose there are U interacting units, indexed by

$$u \in \{1, \ldots, U\},$$

and observation times

$$n \in \{1, \ldots, N\}.$$

For each unit u and time n, let $X_{u,n}$ denote the local latent state and let $Y_{u,n}$ denote the corresponding observation. The full latent state at time n is then

$$\mathbf{X}_n = (X_{1,n}, \ldots, X_{U,n}),$$

and similarly the full observation vector is

$$\mathbf{Y}_n = (Y_{1,n}, \ldots, Y_{U,n}).$$

A spatiotemporal partially observed Markov process, or SpatPOMP model, is simply a POMP model whose latent state is partitioned into interacting units (Asfaw et al., 2021). The interaction enters through the process model, not through an arbitrary relaxation of the measurement structure. Conditional on the latent state, it is often natural to assume that measurements are independent across units,

$$g_n(\mathbf{y}_n \mid \mathbf{x}_n; \theta) = \prod_{u=1}^{U} g_{u,n}(y_{u,n} \mid \mathbf{x}_n; \theta), \tag{2}$$

or, in some applications, that each local observation depends only on a small component of $\mathbf{x}_n$. By contrast, the latent transition typically does not factorize over units, because the units are coupled through migration, contact, transport, or some other interaction mechanism. In epidemiological applications, for example, each unit may be a city, the local latent state may include compartment counts such as susceptible, exposed, infectious, and recovered individuals, and the transition mechanism may allow infection pressure from one city to influence another.

The importance of the SpatPOMP formulation is that it separates two distinct sources of complexity. First, the temporal dynamics are nonlinear and stochastic, as in ordinary POMP models. Second, the state dimension grows with the number of units, and the units are only weakly localized through the process model. This is exactly the regime in which filtering becomes difficult. When U is small, simulation based methods can often approximate the filtering distribution accurately. When U is moderate or large, however, the product structure in the observation density and the coupling in the latent transition interact to produce severe Monte Carlo instability.

A central practical feature of many POMP and SpatPOMP models is that the latent transition density is *implicit*. By this we mean that, given the current state X_{n-1} or $\mathbf{X}_{n-1}$, we can simulate the next state from the process model, but we cannot evaluate the density

$$f_{X_n|X_{n-1}}(x_n \mid x_{n-1}; \theta)$$

in closed form. This is the situation for many mechanistic models defined through stochastic differential equations, Markov jump processes, compartment models with demographic noise, and simulation-first biological models (Bretó et al., 2009). In software such as `pomp` and `spatPomp`, this distinction appears explicitly. The user supplies an `rprocess` simulator for the latent process and a `dmeasure` evaluator for the observation density, but is not required to provide an evaluator for the latent transition density (Asfaw et al., 2024). This plug-and-play property is one of the main reasons particle filtering is so attractive in scientific applications. It allows the inferential methodology to match the way mechanistic models are actually built, namely by specifying how the system evolves and how it is observed, rather than by deriving closed-form transition kernels.

2.2 Particle Filtering and Sequential Monte Carlo

Particle filtering or sequential Monte Carlo is a simulation based family of methods for approximating the filtering distributions

$$p_\theta(\mathbf{x}_n \mid \mathbf{y}_{1:n}), \qquad n = 1, \ldots, N,$$

and the likelihood factors in (1) (Doucet et al., 2001). The basic idea is to represent the filtering distribution at time n by a weighted cloud of particles

$$\left\{\mathbf{X}_n^{(j)}, W_n^{(j)}\right\}_{j=1}^{J},$$

where J is the number of particles. In the bootstrap particle filter, the particles are propagated directly from the process model and then reweighted by the measurement density. This gives a particularly simple algorithm that is fully plug-and-play.

Suppose that after assimilating data through time $n-1$ we have particles

$$\left\{\mathbf{X}_{n-1}^{(j)}\right\}_{j=1}^{J}$$

approximately distributed according to the filtering distribution

$$p_\theta(\mathbf{x}_{n-1} \mid \mathbf{y}_{1:n-1}).$$

The bootstrap particle filter proceeds in three steps. First, each particle is propagated through the latent simulator,

$$\mathbf{X}_n^{P,(j)} \sim f_{\mathbf{X}_n|\mathbf{X}_{n-1}}(\,\cdot \mid \mathbf{X}_{n-1}^{(j)}; \theta), \qquad j = 1, \ldots, J.$$

Second, the propagated particles are assigned measurement weights

$$\widetilde{W}_n^{(j)} = g_n(\mathbf{y}_n \mid \mathbf{X}_n^{P,(j)}; \theta).$$

Third, the weights are normalized,

$$W_n^{(j)} = \frac{\widetilde{W}_n^{(j)}}{\sum_{k=1}^{J} \widetilde{W}_n^{(k)}},$$

and the particles are resampled so that particles with large weight are duplicated and particles with negligible weight are discarded. The resampled particles then serve as the filtering approximation at time n.

The bootstrap particle filter also produces an estimator of the likelihood. At each time n, the predictive likelihood factor in (1) is approximated by the average unnormalized weight,

$$\widehat{p}_\theta(\mathbf{y}_n \mid \mathbf{y}_{1:n-1}) = \frac{1}{J}\sum_{j=1}^{J} \widetilde{W}_n^{(j)}.$$

Therefore the full likelihood estimator is

$$\widehat{p}_\theta(\mathbf{y}_{1:N}) = \prod_{n=1}^{N} \left(\frac{1}{J}\sum_{j=1}^{J} \widetilde{W}_n^{(j)} \right), \tag{3}$$

and the corresponding log likelihood estimator is

$$\widehat{\ell}(\theta) = \sum_{n=1}^{N} \log \left(\frac{1}{J}\sum_{j=1}^{J} \widetilde{W}_n^{(j)} \right). \tag{4}$$

This quantity is central in likelihood based inference for POMP models, because it can be plugged into optimization, profiling, or Monte Carlo adjusted likelihood procedures (King et al., 2016)

Algorithm 1 Bootstrap particle filter for a SpatPOMP model

Initialize particles $\mathbf{X}_0^{(j)} \sim p_\theta(\mathbf{x}_0)$ for $j = 1, \dots, J$
for $n = 1, \dots, N$ **do**
  for $j = 1, \dots, J$ **do**
    Propagate

$$\mathbf{X}_n^{P,(j)} \sim f_{\mathbf{X}_n|\mathbf{X}_{n-1}}(\cdot \mid \mathbf{X}_{n-1}^{(j)}; \theta)$$

    Compute unnormalized weight

$$\widetilde{W}_n^{(j)} = g_n(\mathbf{y}_n \mid \mathbf{X}_n^{P,(j)}; \theta)$$

  end for
  Estimate predictive likelihood

$$\widehat{p}_\theta(\mathbf{y}_n \mid \mathbf{y}_{1:n-1}) = \frac{1}{J} \sum_{j=1}^{J} \widetilde{W}_n^{(j)}$$

  Normalize weights and resample particles

$$W_n^{(j)} = \widetilde{W}_n^{(j)} \Big/ \sum_{k=1}^{J} \widetilde{W}_n^{(k)}$$

  Set $\mathbf{X}_n^{(j)}$ equal to a resampled copy of $\mathbf{X}_n^{P,(j)}$
end for
Return filtering particles and

$$\widehat{\ell}(\theta) = \sum_{n=1}^{N} \log \left(\frac{1}{J} \sum_{j=1}^{J} \widetilde{W}_n^{(j)} \right)$$

For concreteness, Algorithm 1 gives pseudocode for the bootstrap particle filter in the spatiotemporal setting considered here.

The statistical appeal of this algorithm is that it remains valid in settings where the latent process is nonlinear, non-Gaussian, and specified only through simulation (Gordon et al., 1993; Asfaw et al., 2021) Classical theory has established consistency and central limit theorems for particle filter estimators under suitable conditions (Chopin, 2004; Moral, 2004). In applications, however, asymptotic validity is only part of the story. What matters equally is whether a practically affordable number of particles is enough to keep the filter numerically stable.

A useful diagnostic is the effective sample size,

$$\mathrm{ESS}_n = \frac{\left(\sum_{j=1}^{J} \widetilde{W}_n^{(j)} \right)^2}{\sum_{j=1}^{J} \left(\widetilde{W}_n^{(j)} \right)^2}, \tag{5}$$

or, equivalently, if the weights are normalized,

$$\mathrm{ESS}_n = \frac{1}{\sum_{j=1}^{J} (W_n^{(j)})^2}.$$

When the normalized weights are nearly uniform, the ESS is close to J. When one or a few particles dominate, the ESS collapses toward one (Kong et al., 1994). In low dimensional problems the filter often maintains a reasonably large ESS with moderate J. In high dimensional spatiotemporal systems, by contrast, it is common for the ESS to fall sharply at each observation time, leaving the filter with very little diversity after resampling. The central challenge is therefore not merely to simulate particles, but to ensure that enough particles remain relevant once the data are incorporated.

2.3 The Curse of Dimensionality in Filtering

The curse of dimensionality in particle filtering is best understood through the structure of the weights. In a spatiotemporal model with U units, a common observation model takes the form

$$g_n(\mathbf{y}_n \mid \mathbf{x}_n; \theta) = \prod_{u=1}^{U} g_{u,n}(y_{u,n} \mid \mathbf{x}_n; \theta),$$

or at least approximately factors in this way after conditioning on the latent state. Consequently, the unnormalized weight of particle j at time n is

$$\widetilde{W}_n^{(j)} = \prod_{u=1}^{U} g_{u,n}(y_{u,n} \mid \mathbf{X}_n^{P,(j)}; \theta).$$

Taking logarithms gives

$$\log \widetilde{W}_n^{(j)} = \sum_{u=1}^{U} \log g_{u,n}(y_{u,n} \mid \mathbf{X}_n^{P,(j)}; \theta). \tag{6}$$

Even if each unit contributes only moderate variability, the sum in (6) can become highly dispersed when U is large. In effect, small mismatches across many units accumulate into very large differences in total weight. The result is that most particles receive astronomically small weight while a tiny fraction dominate the normalized distribution.

This phenomenon has been analyzed both heuristically and rigorously in the filtering literature. Bengtsson et al. (2008) show how particle filters can collapse in very large systems, and Snyder et al. (2008) give influential arguments that the number of particles required for stable performance can grow exponentially with dimension. Related analyses and extensions appear in later work, including Rebeschini and Van Handel (2015), which also motivates localized filtering approximations in weakly coupled systems (Snyder et al., 2008; Bengtsson et al., 2008; Rebeschini and Van Handel, 2015). The underlying message is not that particle filtering is impossible in high dimensions, but rather that a naive global weighting strategy becomes increasingly ineffective as the observation dimension grows.

For spatiotemporal models, the dimensional burden is tied directly to the number of units. Increasing the number of cities, regions, or spatial locations increases the dimension of the latent state, often increases the amount of data assimilated at each time, and introduces additional coupling in the latent process. The resulting challenge is therefore not simply a matter of adding more variables. Each additional unit contributes its own local uncertainty while also changing the transition law for the entire coupled system. In practical terms, one observes that a standard bootstrap particle filter may perform adequately for small U, then deteriorate rapidly once a threshold is crossed. This deterioration shows up empirically as highly variable log likelihood estimates, repeated particle impoverishment after resampling, and strong sensitivity to the random seed.

The structure of the measles model studied later in this thesis provides a useful concrete example. Each city contributes local infection dynamics, demographic stochasticity, and noisy case reports, while weak interactions between cities affect the evolution of the full latent state. Because the observation model contributes information at every city and every reporting time, each new observation can sharply reduce the number of particles that remain consistent with the data. This is precisely the setting in which one seeks filtering algorithms that respect the plug-and-play structure of the model while using information more gradually than the bootstrap particle filter does.

2.4 Guided Intermediate Resampling Filters (GIRF)

The bootstrap particle filter described in Section 2.2 updates particle weights only when a new observation arrives. This can be problematic when the observation interval is long relative to the local time scale of the latent process, or when the state dimension is large enough that most particles have already drifted away from regions of high observational likelihood by the time the next measurement is processed. Guided intermediate resampling filters which abbreviated as GIRF address this issue by introducing additional resampling times between observation times (Park and Ionides, 2020). The basic idea is simple. Instead of waiting until time t_n to discover that a particle is inconsistent with y_n, one tries to assess this inconsistency gradually over the interval (t_{n-1}, t_n).

To formalize this idea, suppose the observation times are

$$0 = t_0 < t_1 < \cdots < t_N.$$

For a fixed observation interval (t_{n-1}, t_n), choose an integer

$$S = N_{\text{inter}} \geq 1,$$

and define intermediate times

$$t_{n,0} = t_{n-1} < t_{n,1} < \cdots < t_{n,S} = t_n.$$

The quantity S controls how finely the interval is subdivided. When $S = 1$, there are no truly intermediate points and one recovers a standard observation-time update. When $S > 1$, the

filter will simulate and potentially resample several times before reaching t_n. It is convenient to denote the latent state at intermediate time $t_{n,s}$ by

$$\mathbf{X}_{n,s} = \mathbf{X}(t_{n,s}), \qquad s = 0, \ldots, S.$$

The goal is to use the next observation, or more generally a short block of future observations, to evaluate how promising each particle is at these intermediate times.

The central object in GIRF is therefore a *guide function*. For the simplest one-step lookahead setting, one introduces functions

$$u_{n,s}(\mathbf{x}) \approx p_\theta(\mathbf{y}_n \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1}), \qquad s = 0, \ldots, S,$$

with the convention that the final guide at the observation time is chosen to equal the exact measurement density,

$$u_{n,S}(\mathbf{x}) = g_n(\mathbf{y}_n \mid \mathbf{x}; \theta).$$

More generally, one may use a lookahead horizon $L \geq 1$ and define

$$u_{n,s}(\mathbf{x}) \approx p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1}),$$

but in this thesis the main focus is on $L = 1$. The practical role of the guide is to reweight particles at intermediate times according to their compatibility with future observations. A particle whose current state makes the next observation highly unlikely should be down-weighted before reaching the observation time, rather than after all computational effort has already been spent propagating it forward.

The logic of GIRF may be understood through a telescoping product. Suppose that after resampling at stage $(n, s-1)$ the particles approximately represent a distribution proportional to

$$u_{n,s-1}(\mathbf{x})\, p_\theta(\mathbf{x}_{n,s-1} = \mathbf{x} \mid \mathbf{y}_{1:n-1}).$$

If these particles are propagated through the latent process from $t_{n,s-1}$ to $t_{n,s}$, then a natural incremental importance weight is

$$w_{n,s}^{(j)} \propto \frac{u_{n,s}\left(\mathbf{X}_{n,s}^{(j)}\right)}{u_{n,s-1}\left(\mathbf{X}_{n,s-1}^{(a(j))}\right)}, \tag{7}$$

where $a(j)$ denotes the ancestor of particle j after the previous resampling step. Over the full interval, these ratios telescope heuristically to

$$\prod_{s=1}^{S} \frac{u_{n,s}(\mathbf{X}_{n,s})}{u_{n,s-1}(\mathbf{X}_{n,s-1})} \approx \frac{g_n(\mathbf{y}_n \mid \mathbf{X}_{n,S})}{u_{n,0}(\mathbf{X}_{n,0})}.$$

The exact observation likelihood at time t_n is introduced gradually through intermediate resampling rather than all at once at the end of the interval. The guide affects the numerical efficiency of the filter, not the scientific model itself. In the ideal case where $u_{n,s}$ equals the exact predictive likelihood of the future observation given the intermediate state, the

Algorithm 2 Generic guided intermediate resampling filter over one observation interval

Require: particles $\{\mathbf{X}_{n-1}^{(j)}\}_{j=1}^{J}$ approximating $p_\theta(\mathbf{x}_{n-1} \mid \mathbf{y}_{1:n-1})$
Require: intermediate times $t_{n,0} = t_{n-1} < \cdots < t_{n,S} = t_n$
Require: guide functions $\{u_{n,s}\}_{s=0}^{S}$
for $s = 1, \ldots, S$ **do**
  for $j = 1, \ldots, J$ **do**
    Propagate particle

$$\mathbf{X}_{n,s}^{P,(j)} \sim p_\theta(\mathbf{x}_{n,s} \mid \mathbf{X}_{n,s-1}^{(j)})$$

    Compute guide-based incremental weight

$$\widetilde{W}_{n,s}^{(j)} = \frac{u_{n,s}(\mathbf{X}_{n,s}^{P,(j)})}{u_{n,s-1}(\mathbf{X}_{n,s-1}^{(j)})}$$

  end for
  Normalize $\widetilde{W}_{n,s}^{(j)}$ and resample
  Set resampled particles to $\{\mathbf{X}_{n,s}^{(j)}\}_{j=1}^{J}$
end for
Use $\{\mathbf{X}_{n,S}^{(j)}\}_{j=1}^{J}$ as the filtering approximation at time t_n

variance of the importance weights can be greatly reduced. In practice one constructs an approximation that is informative enough to improve stability while remaining computationally feasible (Park and Ionides, 2020).

Algorithm 2 gives a high-level view of GIRF in the one-step lookahead setting. This generic description suppresses several implementation details. In practice the exact form of the weight update can include discounting, stabilization, or special treatment of the final observation step (Park and Ionides, 2020). While the essential principle remains the same. GIRF replaces one abrupt update at time t_n by a sequence of smaller updates between t_{n-1} and t_n. In doing so, it seeks to maintain a larger and more useful particle population in situations where a standard particle filter would collapse.

Two tuning parameters deserve special mention. The first is N_{inter}, the number of intermediate subintervals, which controls how gradually future information is introduced. Larger values can improve stability but also increase computational cost. The second is the lookahead horizon L. A larger L allows the guide to account for a longer future observation block, but this also makes guide construction more demanding. In the present thesis the main empirical focus is on $L = 1$, where the guide only needs to score the compatibility of the current intermediate state with the next observation.

2.4.1 Bootstrap GIRF

The most direct way to build a guide is to estimate the predictive likelihood by Monte Carlo simulation. Suppose one has a particle at intermediate time $t_{n,s}$ with state $\mathbf{x}$. One may

simulate M independent forecast trajectories from $t_{n,s}$ to the next observation time t_n,

$$\mathbf{X}_n^{(m)} \sim p_\theta(\mathbf{x}_n \mid \mathbf{X}_{n,s} = \mathbf{x}), \qquad m = 1, \dots, M,$$

and then estimate the predictive likelihood by

$$\widehat{u}_{n,s}^{\text{boot}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} g_n(\mathbf{y}_n \mid \mathbf{X}_n^{(m)}; \theta). \tag{8}$$

This construction is natural because it directly mirrors the predictive integral in (1). It is fully plug-and-play, requiring only simulation from the latent process and evaluation of the measurement density. Conceptually, it is the most faithful to the meaning of a guide function. If one could make M arbitrarily large, (8) would converge to the desired predictive likelihood under standard Monte Carlo arguments.

In high dimensional spatiotemporal settings, however, directly averaging the full joint measurement density can be numerically unstable. Since the observation density often factors across units, one practical alternative is to estimate guide contributions unit by unit and then combine them. For example, under the factorization

$$g_n(\mathbf{y}_n \mid \mathbf{x}_n; \theta) = \prod_{u=1}^{U} g_{u,n}(y_{u,n} \mid \mathbf{x}_n; \theta),$$

one may define

$$\widehat{u}_{n,s}^{\text{boot}}(\mathbf{x}) = \prod_{u=1}^{U} \left[\frac{1}{M} \sum_{m=1}^{M} g_{u,n}\Big(y_{u,n} \mid \mathbf{X}_n^{(m)}; \theta\Big) \right]. \tag{9}$$

Equation (9) reduces the variance associated with multiplying many tiny Monte Carlo averages, and is close in spirit to the bootstrap-style guide implemented in software such as `spatPomp`. The cost is that one now makes an approximation to the full joint predictive likelihood by combining separate unit-wise predictive components.

It is useful to distinguish this bootstrap-style guide from a standard bootstrap particle filter. In a standard particle filter, one propagates particles from t_{n-1} directly to t_n, weights them by the exact measurement density at t_n, and resamples only once. Bootstrap GIRF, by contrast, still uses a future-observation guide at intermediate times. The guide itself is built from forward simulations, but the filter resamples multiple times within the interval. Therefore the difference from standard PF is not in whether the process is simulated, both methods are simulation based, but in when and how observational information is introduced.

The bootstrap-style guide has an appealing interpretation and can be accurate when enough guide simulations are used. Its main limitations are computational. Each particle at each intermediate time requires a separate bundle of guide simulations, which can become expensive when the number of particles, units, or intermediate steps is large. In addition, the Monte Carlo noise in the guide estimate can itself destabilize the filter if the number of guide simulations is too small. For this reason, prior work has considered cheaper approximate guide constructions based on moments rather than repeated guide simulation (Park and Ionides, 2020).

2.4.2 Moment Guide GIRF

A moment guide replaces repeated Monte Carlo evaluation of the future observation likelihood by an approximate predictive distribution built from low order moments. The guiding idea is that if one can approximately forecast the mean and variance of the next observation given the current intermediate state, then one can evaluate the next observation under a simple parametric approximation. This is often much cheaper than repeatedly simulating guide trajectories for every particle and every intermediate time (Park and Ionides, 2020).

To describe the construction, suppose that for each unit u the measurement model has conditional mean

$$h_{u,n}(\mathbf{x}_n) = \mathbb{E}_\theta[Y_{u,n} \mid \mathbf{X}_n = \mathbf{x}_n]$$

and conditional variance

$$V_{u,n}^{\text{meas}}(\mathbf{x}_n) = \text{Var}_\theta(Y_{u,n} \mid \mathbf{X}_n = \mathbf{x}_n).$$

Further suppose that there is a deterministic or approximately deterministic forecast map

$$\boldsymbol{\mu}_{n,s}(\mathbf{x}) \approx \mathbb{E}_\theta[\mathbf{X}_n \mid \mathbf{X}_{n,s} = \mathbf{x}],$$

which predicts the mean latent state at the next observation time from the current intermediate state. If only this mean forecast were used, one could approximate the observation distribution at time t_n by centering the measurement model at $\boldsymbol{\mu}_{n,s}(\mathbf{x})$. However, this would ignore the process uncertainty that remains between $t_{n,s}$ and t_n. A simulated-moment guide therefore supplements the forecast mean with an approximation to the forecast process variance.

A convenient formulation is to write the total predictive variance for unit u as

$$V_{u,n,s}^{\text{tot}}(\mathbf{x}) = V_{u,n}^{\text{meas}}\Big(\boldsymbol{\mu}_{n,s}(\mathbf{x})\Big) + V_{u,n,s}^{\text{proc}}(\mathbf{x}), \tag{10}$$

where $V_{u,n,s}^{\text{proc}}(\mathbf{x})$ is an approximation to the process-induced uncertainty in the future observation. In the moment guides used in GIRF implementations, this process variance is often estimated by a small number of guide simulations or by a scaling rule that reflects how much uncertainty remains between $t_{n,s}$ and t_n. One then evaluates the observed $y_{u,n}$ under a simple parametric family, often Gaussian or discretized Gaussian, with mean

$$h_{u,n}\Big(\boldsymbol{\mu}_{n,s}(\mathbf{x})\Big)$$

and variance (10). This leads to a guide of the form

$$u_{n,s}^{\text{mom}}(\mathbf{x}) = \prod_{u=1}^{U} \varphi_{u,n}\Big(y_{u,n};\, h_{u,n}\Big(\boldsymbol{\mu}_{n,s}(\mathbf{x})\Big),\, V_{u,n,s}^{\text{tot}}(\mathbf{x})\Big), \tag{11}$$

where $\varphi_{u,n}$ denotes the approximating observation density or mass function for unit u.

The strength of the moment guide is computational efficiency. Once a suitable forecasting rule and variance approximation are available, the guide can be evaluated quickly for many

particles. This is especially useful when the latent process is continuous time and the cost of repeated forward simulation at each substep would otherwise be substantial. The weakness of the moment guide is that it can be inaccurate precisely in the settings that motivate spatiotemporal particle filtering in the first place. If the latent dynamics are strongly nonlinear, if the future observation distribution is skewed or heavy tailed, or if the observation model is discrete with strong overdispersion, then a low-order moment approximation may not preserve the shape of the true predictive distribution. In such situations a moment guide can still improve stability relative to a standard particle filter, because even a rough forecast may be better than no guidance at all, but it may also introduce a mismatch between the true and approximating predictive likelihood (Park and Ionides, 2020).

The present thesis takes this limitation as a motivation. Both bootstrap and moment guides seek to approximate the same quantity, namely the predictive likelihood of future observations given the current intermediate state. The difference is that bootstrap guides approximate this quantity by repeated simulation, whereas moment guides approximate it by a simplified forecast distribution. A neural guide, introduced in later chapters, can be viewed as a third strategy. It still targets the same predictive quantity, but it learns the approximation from simulated data rather than deriving it from analytic moments or repeated guide simulations.

2.5 Neural Posterior Estimation and Simulation-Based Inference

Simulation-based inference, abbreviated as SBI, refers to a broad class of methods for statistical inference in models where data can be simulated but key probability densities are not available in closed form (Hermans et al., 2020). In the classical setting one samples parameters θ from a prior, simulates synthetic data y from the model, and then uses these simulated pairs to train a neural density estimator. Neural posterior estimation, or NPE, is one important member of this family. In its original form, NPE seeks to learn a conditional density

$$q_\phi(\theta \mid y) \approx p(\theta \mid y)$$

from simulated parameter-data pairs (Papamakarios and Murray, 2016; Greenberg et al., 2019). The key idea is that, even when the exact posterior is unavailable, one can still learn a flexible approximation by repeatedly simulating from the model and fitting a neural network to the resulting conditional structure. Recent work has also demonstrated the usefulness of NPE in stochastic epidemic modeling, where neural conditional density estimators can be trained on simulated outbreak data to calibrate complex mechanistic transmission models (Chatha et al., 2024).

The same simulation-based philosophy can be used more broadly than parameter inference. In the present project, the quantity of interest is not primarily a posterior over parameters, but rather a predictive score that can be used as a guide inside a particle filter. If one can generate pairs

$$(\mathbf{X}_{n,s}, \mathbf{Y}_n)$$

from the model, then one can train a neural procedure to distinguish compatible pairs from incompatible pairs, or directly to approximate a predictive likelihood of future observations

given an intermediate state. From this viewpoint, the role of NPE in this thesis is to provide a learned approximation to the future-data compatibility that GIRF requires at intermediate times. The network itself will be introduced in the methodology chapter. At the background level, it is enough to note that simulation-based inference provides a principled route to learning guide functions when direct derivation is difficult.

One especially useful perspective comes from contrastive objectives, such as InfoNCE, which learn a score function by asking a model to identify which observation belongs with which latent state among a batch of candidates. Suppose one has simulated positive pairs

$$(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), \qquad i = 1, \dots, B,$$

drawn from the joint distribution of intermediate states and future observations. A neural network produces a scalar score

$$s_\phi(\mathbf{x}, \mathbf{y}),$$

and is trained with the loss

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp\Big(s_\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\Big)}{\sum_{j=1}^{B} \exp\Big(s_\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(j)})\Big)}. \tag{12}$$

This objective encourages the score of the true pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ to exceed the scores of mismatched pairs $(\mathbf{x}^{(i)}, \mathbf{y}^{(j)})$ for $j \neq i$. Under idealized conditions, such contrastive objectives recover a score proportional to a log density ratio, schematically

$$s_\phi(\mathbf{x}, \mathbf{y}) \propto \log p(\mathbf{y} \mid \mathbf{x}) - \log p(\mathbf{y}),$$

or equivalently to

$$\log p(\mathbf{x}, \mathbf{y}) - \log p(\mathbf{x}) - \log p(\mathbf{y}),$$

up to additive constants that do not depend on the variable being ranked (Oord et al., 2018; Hermans et al., 2020). For filtering applications this is appealing because particle resampling only depends on relative weights. If, for a fixed future observation $\mathbf{y}_n$, the score $s_\phi(\mathbf{x}, \mathbf{y}_n)$ orders particles in approximately the same way as $\log p_\theta(\mathbf{y}_n \mid \mathbf{x})$, then the score can function as a guide even if it is not a perfectly calibrated likelihood.

This observation explains why neural methods are relevant to guided filtering. GIRF does not require a full posterior over states or parameters at intermediate times. It requires a function that scores each particle according to how plausible the next observation is under that particle. Contrastive simulation-based inference offers one route to learning exactly such a score from synthetic data, while retaining compatibility with implicit latent dynamics. The neural method proposed later in this thesis builds on this principle. For now, the important point is conceptual. Neural posterior estimation and related SBI methods provide a data-driven mechanism for approximating the predictive information needed by guided particle filters, without requiring analytic transition densities or handcrafted local Gaussian approximations.

3 Methodology

3.1 Guide Construction as an Exact Posterior Estimation Problem

The practical challenge in GIRF is not the intermediate resampling loop itself, but the construction of a guide function that is both informative and computationally feasible. In order to clarify the relationship between guide construction and neural training, it is useful to distinguish three different levels of description.

At the first level, GIRF requires an ideal predictive guide. At the second level, this ideal guide can be rewritten exactly, by Bayes' rule, in terms of an intermediate-state posterior distribution. At the third level, the implemented neural method does not directly estimate that posterior, but instead learns a contrastive score that approximates the corresponding posterior-to-prior ratio. This subsection develops the exact reformulation, while the next subsection explains the practical surrogate used in computation.

Fix a parameter value θ, an observation interval n, and an intermediate substep $s \in \{0, \dots, S-1\}$. Recall that for a general lookahead horizon $L \geq 1$, the future observation block is

$$\mathbf{Y}_{n:n+L-1} = (\mathbf{Y}_n, \mathbf{Y}_{n+1}, \dots, \mathbf{Y}_{n+L-1}).$$

The ideal guide at substep s is

$$\psi^{\star}_{n,s}(\mathbf{x}) = p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1})\,. \tag{13}$$

This is the quantity that GIRF would use if the predictive likelihood of the future observation block were available exactly.

Because the latent process is Markov, the future observation block is conditionally independent of the past observations once the current latent state is known. Therefore

$$p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1}) = p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x})\,, \tag{14}$$

and hence the ideal guide can be written more simply as

$$\psi^{\star}_{n,s}(\mathbf{x}) = p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x})\,. \tag{15}$$

Thus the ideal guide is exactly the predictive likelihood of the future observation block given the intermediate latent state.

The next object needed for the reformulation is the predictive distribution of the intermediate state itself. Define

$$\pi_{n,s}(\mathbf{x}) := p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{y}_{1:n-1})\,. \tag{16}$$

This is the runtime reference distribution for guide evaluation. Before the future observation block $\mathbf{y}_{n:n+L-1}$ has been incorporated, the particles used by GIRF are intended to approximate $\pi_{n,s}$. Consequently, $\pi_{n,s}$ is the natural predictive distribution against which any intermediate-state posterior should be compared.

Under this predictive distribution, the joint law of an intermediate state and its associated future observation block is

$$\rho_{n,s}(\mathbf{x}, \mathbf{y}_{n:n+L-1}) = \pi_{n,s}(\mathbf{x})\, p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{x})\,. \tag{17}$$

Equation (17) provides the exact simulation-based target distribution for training. One may first sample $\mathbf{X}_{n,s}$ from its predictive law, and then simulate the future observation block conditional on that state. This is precisely the type of joint distribution required for simulation-based conditional density estimation.

Now define the posterior distribution of the intermediate state given the future observation block,

$$\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) := p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1}, \mathbf{y}_{1:n-1})\,. \tag{18}$$

Suppose that a conditional density estimator

$$q_\phi(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}, n, s)$$

is trained on samples from the joint law (17). The corresponding population negative log-likelihood objective is

$$\mathcal{L}_{\mathrm{NPE}}(\phi) = \mathbb{E}_{(\mathbf{X}_{n,s}, \mathbf{Y}_{n:n+L-1}) \sim \rho_{n,s}} \left[-\log q_\phi(\mathbf{X}_{n,s} \mid \mathbf{Y}_{n:n+L-1}, n, s) \right]. \tag{19}$$

To identify the population minimizer of (19), condition on the future observation block. This gives

$$\mathcal{L}_{\mathrm{NPE}}(\phi) = \mathbb{E}_{\mathbf{Y}_{n:n+L-1} \sim p_\theta(\cdot \mid \mathbf{y}_{1:n-1})} \left[\int \eta_{n,s}(\mathbf{x} \mid \mathbf{Y}_{n:n+L-1}) \left(-\log q_\phi(\mathbf{x} \mid \mathbf{Y}_{n:n+L-1}, n, s) \right) d\mathbf{x} \right].$$

For a fixed realization $\mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1}$, the inner integral is

$$\int \eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) \left(-\log q_\phi(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}, n, s) \right) d\mathbf{x}.$$

Adding and subtracting the entropy of $\eta_{n,s}(\cdot \mid \mathbf{y}_{n:n+L-1})$ yields

$$\begin{aligned} &\int \eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) \left(-\log q_\phi(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}, n, s) \right) d\mathbf{x} \\ &\quad = \mathrm{KL}(\eta_{n,s}(\cdot \mid \mathbf{y}_{n:n+L-1}) \parallel q_\phi(\cdot \mid \mathbf{y}_{n:n+L-1}, n, s)) + H(\eta_{n,s}(\cdot \mid \mathbf{y}_{n:n+L-1}))\,. \end{aligned} \tag{20}$$

The entropy term does not depend on ϕ. Therefore the unique population minimizer of (19) is

$$q_\phi^\star(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}, n, s) = \eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1})\,. \tag{21}$$

Thus, in the exact formulation, training a conditional density estimator on simulated pairs $(\mathbf{X}_{n,s}, \mathbf{Y}_{n:n+L-1})$ is equivalent to learning the posterior distribution of the intermediate state given the future observation block.

The relevance of this posterior to guide construction follows from Bayes' rule. By definition,

$$\begin{aligned} \eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) &= p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1}, \mathbf{y}_{1:n-1}) \\ &= \frac{p_\theta(\mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1})\; p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{y}_{1:n-1})}{p_\theta(\mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1} \mid \mathbf{y}_{1:n-1})}. \end{aligned} \tag{22}$$

Using the definitions of the ideal guide and the predictive intermediate-state distribution, equation (22) becomes

$$\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) = \frac{\psi^{\star}_{n,s}(\mathbf{x})\, \pi_{n,s}(\mathbf{x})}{m_n(\mathbf{y}_{n:n+L-1})}, \tag{23}$$

where

$$m_n(\mathbf{y}_{n:n+L-1}) := p_\theta(\mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1} \mid \mathbf{y}_{1:n-1}) \tag{24}$$

is the predictive marginal distribution of the future observation block. Rearranging (23) gives

$$\psi^{\star}_{n,s}(\mathbf{x}) = \frac{\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1})}{\pi_{n,s}(\mathbf{x})}\, m_n(\mathbf{y}_{n:n+L-1})\,. \tag{25}$$

Equation (25) shows that the ideal guide is equal, up to a multiplicative factor depending only on the future observation block, to the ratio between the intermediate-state posterior and the predictive distribution of that same intermediate state.

This becomes even more relevant in GIRF because the filter uses ratios of guides between successive substeps. For consecutive substeps $s-1$ and s,

$$\begin{aligned}\frac{\psi^{\star}_{n,s}(\mathbf{x}_s)}{\psi^{\star}_{n,s-1}(\mathbf{x}_{s-1})} &= \frac{\eta_{n,s}(\mathbf{x}_s \mid \mathbf{y}_{n:n+L-1})\; m_n(\mathbf{y}_{n:n+L-1})\,/\pi_{n,s}(\mathbf{x}_s)}{\eta_{n,s-1}(\mathbf{x}_{s-1} \mid \mathbf{y}_{n:n+L-1})\; m_n(\mathbf{y}_{n:n+L-1})\,/\pi_{n,s-1}(\mathbf{x}_{s-1})} \\ &= \frac{\eta_{n,s}(\mathbf{x}_s \mid \mathbf{y}_{n:n+L-1})\,/\pi_{n,s}(\mathbf{x}_s)}{\eta_{n,s-1}(\mathbf{x}_{s-1} \mid \mathbf{y}_{n:n+L-1})\,/\pi_{n,s-1}(\mathbf{x}_{s-1})}.\end{aligned} \tag{26}$$

The predictive marginal term $m_n(\mathbf{y}_{n:n+L-1})$ cancels exactly. Therefore the operational quantity in GIRF is not the posterior alone and not the guide alone, but the posterior-to-prior ratio.

Proposition 1 For fixed parameter θ, interval n, and substep s, define

$$\pi_{n,s}(\mathbf{x}) = p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{y}_{1:n-1})\,,$$

$$\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) = p_\theta(\mathbf{X}_{n,s} = \mathbf{x} \mid \mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1}, \mathbf{y}_{1:n-1})\,,$$

and

$$\psi^{\star}_{n,s}(\mathbf{x}) = p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1})\,.$$

Then the ideal guide satisfies

$$\psi^{\star}_{n,s}(\mathbf{x}) = \frac{\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1})}{\pi_{n,s}(\mathbf{x})}\, m_n(\mathbf{y}_{n:n+L-1})\,,$$

where

$$m_n(\mathbf{y}_{n:n+L-1}) = p_\theta(\mathbf{Y}_{n:n+L-1} = \mathbf{y}_{n:n+L-1} \mid \mathbf{y}_{1:n-1})\,.$$

Consequently, constructing the ideal GIRF guide admits an exact reformulation as estimation of the family of intermediate-state posterior distributions

$$\eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1})\,.$$

The proposition gives a precise mathematical interpretation of guide construction. It also clarifies why the state distribution used in training matters. The posterior $\eta_{n,s}$ is defined relative to the predictive distribution $\pi_{n,s}$, and $\pi_{n,s}$ is exactly the runtime intermediate-state distribution on which the guide is evaluated. If the training data are generated from a substantially different state distribution, then even a statistically accurate estimator for the simulated training problem may be poorly aligned with the states actually encountered during filtering.

At the same time, the proposition reveals an important practical obstacle. To recover the ideal guide exactly from (25), one would need both the posterior $\eta_{n,s}$ and the predictive density $\pi_{n,s}(\mathbf{x})$. In implicit spatiotemporal models, $\pi_{n,s}$ is typically available only through simulation and is not analytically tractable. This leads directly to the third and final level of description. The exact posterior reformulation is mathematically valid, but it is not the most convenient computational object to estimate directly.

3.2 Implemented Training Target, Contrastive Ratio Estimation

The exact posterior reformulation above explains what the guide means mathematically, but it is not the object estimated directly in implementation. In the implemented method, the neural network produces a scalar score

$$s_\phi(\mathbf{x}, \mathbf{y}),$$

rather than a normalized conditional density. The score should be large when the intermediate state $\mathbf{x}$ is compatible with the future observation block $\mathbf{y}_{n:n+L-1}$, and small otherwise. For the main experiments in this thesis, $L = 1$, so the future observation block reduces to $\mathbf{y}_n$, but the notation in this subsection remains general.

Training examples consist of simulated positive pairs

$$(\mathbf{x}^{(i)}_{n,s}, \mathbf{y}^{(i)}_{n:n+L-1}), \qquad i = 1, \ldots, B,$$

where B is the batch size. In the implementation, batches are grouped by a common intermediate-step identifier so that all examples in a batch correspond to the same sub-step position. This prevents the network from solving the contrastive task primarily through time encodings, and encourages it to use the latent state itself to distinguish which future observation block belongs with which intermediate state.

For a batch of positive pairs

$$(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}_{n:n+L-1}), \ldots, (\mathbf{x}^{(B)}, \mathbf{y}^{(B)}_{n:n+L-1}),$$

define the similarity matrix

$$S_{ij} = s_\phi\left(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}_{n:n+L-1}\right).$$

The training objective is the InfoNCE loss

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}. \tag{27}$$

The pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}_{n:n+L-1})$ is the positive match in row i, while the pairs $(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}_{n:n+L-1})$ for $j \neq i$ serve as negatives.

Under standard density-ratio arguments, the population optimum of this contrastive objective yields a score satisfying

$$s_\phi(\mathbf{x}, \mathbf{y}_{n:n+L-1}) \approx \log p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{x}) + C(\mathbf{y}_{n:n+L-1}), \tag{28}$$

where $C(\mathbf{y}_{n:n+L-1})$ is a function that does not depend on $\mathbf{x}$. Equivalently, using the exact posterior reformulation from the previous subsection,

$$s_\phi(\mathbf{x}, \mathbf{y}_{n:n+L-1}) \approx \log \eta_{n,s}(\mathbf{x} \mid \mathbf{y}_{n:n+L-1}) - \log \pi_{n,s}(\mathbf{x}), \tag{29}$$

again up to an additive function depending only on $\mathbf{y}_{n:n+L-1}$. Thus the implemented score should be understood not as a direct approximation to the posterior itself, but as an approximation to the posterior-to-prior log ratio. This is precisely the quantity that appears in equation (26) and is therefore sufficient for intermediate resampling.

The neural guide used in filtering is obtained by identifying the learned score with a log guide up to the irrelevant additive term,

$$\log \psi^{\mathrm{NN}}_{n,s}(\mathbf{x}) \approx s_\phi(\mathbf{x}, \mathbf{y}_{n:n+L-1}). \tag{30}$$

For the main implementation with $L = 1$, this becomes

$$\log \psi^{\mathrm{NN}}_{n,s}(\mathbf{x}) \approx s_\phi(\mathbf{x}, \mathbf{y}_n).$$

The earlier proposition explains the exact Bayesian quantity of interest. The contrastive score in (30) explains the implemented surrogate. These are consistent, but they are not identical, and making this distinction explicit helps avoid confusion.

The score function includes a temperature parameter $\tau > 0$. In implementation,

$$s_\phi(\mathbf{x}, \mathbf{y}) = \frac{\left\langle z^X_\phi(\mathbf{x}), z^Y_\phi(\mathbf{y}) \right\rangle}{\tau}, \tag{31}$$

where z^X_ϕ and z^Y_ϕ are normalized embeddings of the latent state and future observation block. Small values of τ sharpen the softmax in (27) and can improve training discrimination, but they can also produce guide scores that are too extreme when inserted into a particle filter. Large values flatten the score and can reduce guide usefulness. For this reason, the implementation constrains τ to remain in a prescribed interval, and in some settings fixes or lower-bounds it to improve downstream filtering stability.

3.3 Simulation-Based Data Generation and Amortization

The training data are generated entirely from the mechanistic model. For each simulated trajectory and each observation interval, the interval is subdivided according to the same intermediate grid used by GIRF. At each retained substep s, the intermediate latent state

$\mathbf{X}_{n,s}$ is paired with the future observation block $\mathbf{Y}_{n:n+L-1}$. In this way, the training set approximates samples from the joint law

$$\rho_{n,s}(\mathbf{x}, \mathbf{y}_{n:n+L-1}) = \pi_{n,s}(\mathbf{x})\, p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{x}),$$

which is exactly the joint law underlying the posterior reformulation above.

This design also explains the amortized character of the method. The neural network is trained once on a large collection of simulated pairs and then reused across many intervals, particles, and filtering runs. The resulting guide is therefore an amortized approximation to the ideal predictive guide. Unlike a hand-crafted moment guide, it learns the relevant relationship directly from simulations. Unlike an online bootstrap guide, it does not require repeated guide simulation during every filtering run.

In low-dimensional settings it may be feasible to retain every intermediate substep from every interval of every simulation. In higher-dimensional settings this can become computationally burdensome, because the number of training pairs scales with the number of simulations, the number of observation intervals, the number of retained substeps, and the number of units. The methodology therefore permits retaining only a subset of representative substeps when necessary. This does not change the definition of the method. It is simply a practical approximation to the family of runtime distributions $\pi_{n,s}$ used in guide evaluation.

The next chapter specializes this general methodology to the measles SpatPOMP model and describes the concrete experimental pipeline used to compare bootstrap-guide GIRF, moment-guide GIRF, and the proposed neural-guided variant.

3.4 Neural Guided Intermediate Resampling Filter

The previous subsections separated three distinct but closely related objects. First, GIRF requires an ideal guide

$$\psi^\star_{n,s}(\mathbf{x}) = p_\theta(\mathbf{y}_{n:n+L-1} \mid \mathbf{X}_{n,s} = \mathbf{x}, \mathbf{y}_{1:n-1}),$$

which is the predictive likelihood of a future observation block given the current intermediate state. Second, by Bayes' rule, this ideal guide can be written exactly in terms of the intermediate-state posterior and the predictive intermediate-state distribution. Third, the implemented neural method does not estimate that posterior explicitly, but instead learns a contrastive score

$$s_\phi(\mathbf{x}, \mathbf{y})$$

that approximates the corresponding posterior-to-prior log ratio, and hence serves as a numerically useful surrogate for the log guide. The neural guided intermediate resampling filter, abbreviated as NN-GIRF, is obtained by replacing the hand-constructed intermediate guide in GIRF with this learned score, while leaving the propagation, exact measurement evaluation at observation times, and likelihood accumulation logic otherwise unchanged.

At a high level, NN-GIRF preserves the entire GIRF filtering skeleton. Particles are initialized at the previous observation time, propagated across a grid of intermediate times,

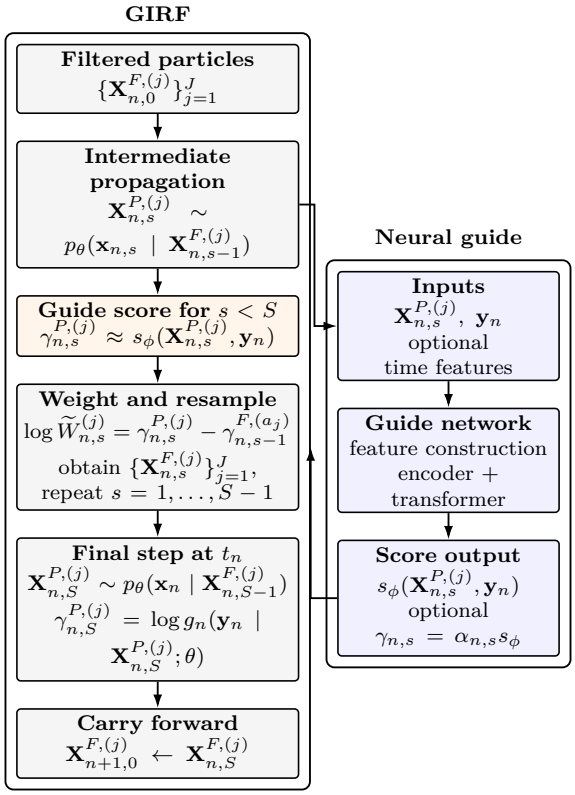


Figure 1: Compact NN-GIRF schematic. The learned guide is used only for intermediate steps $s < S$; the exact observation likelihood is still evaluated at $s = S$.

reweighted and resampled at each substep, and finally updated using the exact observation density at the end of the interval. The only approximation enters through the intermediate guide evaluation. At each substep $s < S$, the proposed particle $\mathbf{X}_{n,s}^{P,(j)}$ is scored against the future observation block by a trained neural network. This produces a particle-specific log guide value, which is then compared against the stored guide value from the previous substep to form an incremental weight ratio. At the final substep $s = S$, the approximate guide is replaced by the exact measurement density, so that the observation-time contribution remains exact. This separation is important. The latent process simulator and the measurement model are not changed, and the approximation is localized entirely to the intermediate guide.

Figure 1 is intended to summarize this logic graphically. The figure will show one observation interval, the intermediate propagation and resampling loop, and the place where the neural network is queried to evaluate guide scores for the future observation block. It will also highlight that the observation-time update still uses the exact $d$measure calculation.

To keep the exposition aligned with the experiments in this thesis, Algorithm 3 presents the direct-score implementation used in the main results. This is the $L = 1$ specialization with a single neural guide evaluation per particle and per intermediate substep. A more general forecast-ensemble extension, which allows $K > 1$ pseudo-guide realizations and a deterministic skeleton μ, is described after the main algorithm.

Algorithm 3 NN-GIRF, direct-score implementation

Require: Initial state simulator $p_\theta(\mathbf{x}_0)$, latent process simulator $p_\theta(\mathbf{x}_{n,s} \mid \mathbf{x}_{n,s-1})$, measurement density $g_n(\mathbf{y}_n \mid \mathbf{x}; \theta)$

Require: Trained neural score $s_\phi(\mathbf{x}, \mathbf{y})$, particle count J, intermediate count $S = N_{\text{inter}}$, discount schedule $\alpha_{n,s}$, data $\mathbf{y}_{1:N}$

Initialize filtered particles

$$\mathbf{X}_{1,0}^{F,(j)} \sim p_\theta(\mathbf{x}_0), \qquad \gamma_{0,S}^{F,(j)} \leftarrow 0, \qquad j = 1, \dots, J$$

▷ $\gamma_{n,s}^{F,(j)}$ stores the filtered log-guide value carried by particle j

for $n = 1, \dots, N$ **do**

  Define intermediate grid

$$t_{n,0} = t_{n-1} < t_{n,1} < \cdots < t_{n,S} = t_n$$

  for $s = 1, \dots, S$ **do**

    for $j = 1, \dots, J$ **do**

      Propagate particle

$$\mathbf{X}_{n,s}^{P,(j)} \sim p_\theta(\mathbf{x}_{n,s} \mid \mathbf{X}_{n,s-1}^{F,(j)})$$

      if $s < S$ **then**

        Evaluate neural log guide

$$\gamma_{n,s}^{P,(j)} \leftarrow \alpha_{n,s}\, s_\phi\Big(\mathbf{X}_{n,s}^{P,(j)}, \mathbf{y}_n\Big)$$

      else

        Evaluate exact observation-time log guide

$$\gamma_{n,S}^{P,(j)} \leftarrow \log g_n\Big(\mathbf{y}_n \mid \mathbf{X}_{n,S}^{P,(j)}; \theta\Big)$$

      end if

    end for

    for $j = 1, \dots, J$ **do**

      if $s = 1$ and $n \geq 2$ **then**

        Apply observation-time reset

$$\log \widetilde{W}_{n,1}^{(j)} \leftarrow \log g_{n-1}\Big(\mathbf{y}_{n-1} \mid \mathbf{X}_{n,0}^{F,(j)}; \theta\Big) + \gamma_{n,1}^{P,(j)} - \gamma_{n-1,S}^{F,(j)}$$

      else

        Compute log guide-ratio weight

$$\log \widetilde{W}_{n,s}^{(j)} \leftarrow \gamma_{n,s}^{P,(j)} - \gamma_{n,s-1}^{F,(j)}$$

      end if

    end for

Compute stable log-likelihood increment

$$m_{n,s} \leftarrow \max_{1\le j\le J} \log \widetilde{W}_{n,s}^{(j)}$$

$$c_{n,s} \leftarrow m_{n,s} + \log\left(\frac{1}{J}\sum_{j=1}^{J} \exp\left(\log \widetilde{W}_{n,s}^{(j)} - m_{n,s}\right)\right)$$

Normalize weights

$$\overline{W}_{n,s}^{(j)} \leftarrow \frac{\exp(\log \widetilde{W}_{n,s}^{(j)} - m_{n,s})}{\sum_{q=1}^{J} \exp(\log \widetilde{W}_{n,s}^{(q)} - m_{n,s})}$$

Resample ancestor indices $r_1, \ldots, r_J$ with

$$\mathbb{P}(r_j = q) = \overline{W}_{n,s}^{(q)}$$

for $j = 1, \ldots, J$ **do**

Propagate ancestry

$$\mathbf{X}_{n,s}^{F,(j)} \leftarrow \mathbf{X}_{n,s}^{P,(r_j)}$$

$$\gamma_{n,s}^{F,(j)} \leftarrow \gamma_{n,s}^{P,(r_j)}$$

end for

end for

if $n < N$ **then**

for $j = 1, \ldots, J$ **do**

Carry filtered particles to the next interval

$$\mathbf{X}_{n+1,0}^{F,(j)} \leftarrow \mathbf{X}_{n,S}^{F,(j)}$$

end for

end if

end for

return

$$\widehat{\ell}^{\text{NN-GIRF}}(\theta) = \sum_{n=1}^{N}\sum_{s=1}^{S} c_{n,s}$$

and final filtered particles $\{\mathbf{X}_{N,S}^{F,(j)}\}_{j=1}^{J}$

Algorithm 3 makes explicit that NN-GIRF modifies only the guide-evaluation stage. The state propagation remains a standard plug-and-play simulation from the latent process, and the exact measurement density is still used at the observation time. The neural component enters only through the intermediate log guide $\gamma_{n,s}^{P,(j)}$ at $s < S$. The reset step at the start of each new interval is necessary because the final guide value from the previous interval already equals the exact observation-time log likelihood. Without this correction, the denominator in the guide ratio would incorrectly divide out observational information that has already been incorporated into the filtered particle system.

The direct-score algorithm above is the version used in the empirical work of this thesis. Nevertheless, it is useful to record a more general forecast-ensemble extension, since it clarifies how one could combine the present neural guide with pseudo-guide trajectories generated around a deterministic skeleton. Let $\mu(\cdot)$ denote a deterministic forecast map and let K be the number of pseudo-guide realizations. For each propagated particle $\mathbf{X}_{n,s}^{P,(j)}$, one may generate pseudo-guide states

$$\widehat{\mathbf{X}}_{n,s,\ell}^{(j,k)}, \qquad k = 1, \ldots, K, \qquad \ell \in \mathbb{L}_n,$$

where $\mathbb{L}_n = \{n, \ldots, \min(n + L - 1, N)\}$ is the lookahead index set. The neural guide can then be aggregated on the log scale by

$$\gamma_{n,s}^{P,(j)} = \log \left[\frac{1}{K} \sum_{k=1}^{K} \exp \left\{ \sum_{\ell \in \mathbb{L}_n} \alpha_{n,s,\ell} \, s_\phi\Big(\widehat{\mathbf{X}}_{n,s,\ell}^{(j,k)}, \mathbf{y}_\ell\Big) \right\} \right]. \tag{32}$$

Equation (32) reduces to the direct-score implementation when $K = 1$ and $L = 1$. It is therefore best viewed as a generalization of the same neural guiding principle rather than as a separate algorithm.

The neural guided filter can now be summarized in one sentence. GIRF specifies where and how guide values enter the filtering recursion. The exact posterior reformulation explains what the ideal guide means mathematically. The implemented contrastive score $s_\phi(\mathbf{x}, \mathbf{y})$ provides a tractable surrogate for this quantity. NN-GIRF is simply the combination of these three pieces into a single plug-and-play filtering algorithm.

4 Experiments

4.1 Experimental Setup

The experiments in this thesis are designed to compare filtering methods under a common simulation and evaluation protocol. Throughout, the same spatiotemporal partially observed Markov process model is used, the same observation sequence is generated or reused across competing methods, and only the filtering algorithm and its tuning parameters are varied. In particular, when comparing bootstrap-guided intermediate resampling, moment-guided intermediate resampling, and neural-guided intermediate resampling, the latent process model, the measurement model, the number of units, the number of observation times, and the random seed are matched whenever this is meaningful. This design ensures that performance differences can be attributed as directly as possible to the guide construction and filtering strategy rather than to changes in the underlying data-generating mechanism.

For each experiment, the main accuracy metric is the estimated log likelihood normalized by the number of units and the number of observation times,

$$\frac{\widehat{\ell}(\theta)}{UN},$$

so that systems of different sizes can be compared on a common scale. In addition to likelihood-based comparisons, wall-clock runtime, Monte Carlo variability across replicates, and diagnostic quantities such as effective sample size proxies and unit-by-time likelihood summaries are recorded whenever available. In higher-dimensional settings, memory usage is also of interest, since some guide constructions require storing or simulating additional forecast objects whose size grows quickly with the number of units and intermediate steps.

The primary empirical model in this thesis is a spatiotemporal measles model implemented in `spatPomp` through the `measles()` constructor (He et al., 2010; Park and Ionides, 2020; Asfaw et al., 2024). This model is particularly well suited to the present study because it combines several features that make filtering difficult in practice. The latent dynamics are nonlinear and stochastic, the observation model is count-valued and overdispersed, and the state dimension increases directly with the number of cities under study. At the same time, the model is scientifically interpretable and has served as a benchmark for spatiotemporal filtering methods in previous work. The experiments therefore focus on how the three GIRF variants behave as the number of cities and the filtering horizon increase, and on whether a learned neural guide can improve numerical stability in settings where hand-designed guides perform poorly.

4.2 The Measles SpatPOMP Model

The measles model describes transmission within and between multiple cities, with each city treated as a single epidemiological unit. Within a city, the population is partitioned into susceptible, exposed, infectious, and recovered classes. Infection occurs locally through contact between susceptible and infectious individuals, while movement between cities introduces weak but epidemiologically important spatial coupling. In plain language, large cities can sustain repeated outbreaks because they contain enough infectious individuals to maintain transmission, whereas smaller cities often experience local fadeout and are re-seeded through imported infection from elsewhere. This combination of local stochasticity and weak spatial coupling makes the model both scientifically interesting and computationally challenging. The observations are reported measles case counts aggregated over two-week intervals, so each observation represents a noisy and under-reported summary of the underlying infection and removal dynamics over a short reporting window rather than a direct measurement of the latent state itself (Asfaw et al., 2024; Park and Ionides, 2020).

Formally, let

$$S_u(t), \quad E_u(t), \quad I_u(t), \quad R_u(t), \qquad u = 1, \dots, U,$$

denote the numbers of susceptible, exposed, infectious, and recovered individuals in city u at continuous time t. The latent state is

$$\mathbf{X}(t) = \Big(X_1(t), \dots, X_U(t)\Big), \qquad X_u(t) = \Big(S_u(t), E_u(t), I_u(t), R_u(t)\Big).$$

The process dynamics are written in terms of counting processes for transitions between compartments. Following the notation used in the `spatPomp` tutorial, the within-city dynamics

can be summarized as

$$\begin{aligned} dS_u(t) &= dN_{BS,u}(t) - dN_{SE,u}(t) - dN_{SD,u}(t), \\ dE_u(t) &= dN_{SE,u}(t) - dN_{EI,u}(t) - dN_{ED,u}(t), \qquad u = 1, \dots, U, \\ dI_u(t) &= dN_{EI,u}(t) - dN_{IR,u}(t) - dN_{ID,u}(t), \end{aligned} \tag{33}$$

with the recovered population defined implicitly through the known total population,

$$P_u(t) = S_u(t) + E_u(t) + I_u(t) + R_u(t).$$

Here $N_{BS,u}(t)$ represents recruitment into the susceptible class, primarily through births, $N_{SE,u}(t)$ records new infections, $N_{EI,u}(t)$ records progression from exposed to infectious, and $N_{IR,u}(t)$ records progression from infectious to recovered or removed. The terms $N_{\bullet D,u}(t)$ represent outflow from each compartment, primarily through death or emigration. In the implementation used in this thesis, these flows are simulated in `rprocess`, so the model is specified through a simulator for the latent Markov process rather than through an analytically tractable transition density.

The force of infection in city u combines local transmission with a spatial coupling term. In the general form described in the tutorial, the expected increment of the infection counting process over a short interval dt is

$$\begin{aligned} \mathbb{E}[N_{SE,u}(t+dt) - N_{SE,u}(t)] = \beta(t)\, S_u(t) \Bigg[& \left(\frac{I_u(t) + \iota}{P_u(t)} \right)^{\alpha} \\ & + \sum_{\tilde{u} \neq u} \frac{v_{u\tilde{u}}}{P_u(t)} \left\{ \left(\frac{I_{\tilde{u}}(t)}{P_{\tilde{u}}(t)} \right)^{\alpha} - \left(\frac{I_u(t)}{P_u(t)} \right)^{\alpha} \right\} \Bigg] dt + o(dt). \end{aligned} \tag{34}$$

The parameter $\beta(t)$ captures school-term seasonality, α is a mixing exponent, and ι represents imported infection from outside the modeled system. The spatial coupling term depends on travel between cities. In the model family used here, travel rates follow a gravity construction of the form

$$v_{u\tilde{u}} = g\, V_{u\tilde{u}},$$

where g is a global coupling parameter and $V_{u\tilde{u}}$ is a fixed matrix derived from city populations and pairwise distances. This structure implies that the latent transition does not factor across cities even though the observations are conditionally independent given the latent state. The transition from exposed to infectious and from infectious to removed are modeled as conditionally Poisson processes with rates usually denoted by σ and γ, and the latent process therefore remains fully plug-and-play, since these mechanisms are implemented by simulation in continuous time.

The measurement model translates latent epidemic activity into reported case counts at discrete observation times. Let the observation times be

$$0 = t_0 < t_1 < \cdots < t_N,$$

with $t_n - t_{n-1} = 1/26$ year, corresponding to biweekly reporting. For city u, define the latent number of removals in the nth reporting interval by

$$Z_{u,n} = N_{IR,u}(t_n) - N_{IR,u}(t_{n-1}).$$

The observed case count is then a noisy, under-reported version of this latent interval total. In the formulation used in `measles()` and described in the tutorial, the reported cases $Y_{u,n}$ are modeled as a discretized conditionally Gaussian random variable with mean

$$\mathbb{E}[Y_{u,n} \mid Z_{u,n} = z] = \rho z$$

and variance

$$\mathrm{Var}(Y_{u,n} \mid Z_{u,n} = z) = \rho(1-\rho)z + \psi^2\rho^2 z^2.$$

For $y > 0$, this gives

$$\begin{aligned}\mathbb{P}(Y_{u,n} = y \mid Z_{u,n} = z) = \Phi\Big(y + 0.5;\, \rho z,\, \rho(1-\rho)z + \psi^2\rho^2 z^2\Big) \\ - \Phi\Big(y - 0.5;\, \rho z,\, \rho(1-\rho)z + \psi^2\rho^2 z^2\Big),\end{aligned} \tag{35}$$

with the lower limit replaced by $-\infty$ when $y = 0$. In the software implementation this likelihood appears through `dmeasure`. It is this observation model that enters both the exact weighting step at observation times and the guide constructions that attempt to predict future observations from intermediate latent states.

All experiments in this thesis begin from a larger simulated measles object and then subset both the number of cities and the number of observation times. If the full simulation contains $U_{\max}$ cities and $N_{\max}$ observation times, then a smaller test problem with U cities and N observations is obtained by taking the first U units and the first N observation times. This construction is convenient for controlled scaling experiments because the scientific model, parameterization, and code path remain fixed while the effective dimension changes. Increasing U increases the dimension of the latent state, the number of observation components assimilated at each time, and typically the computational difficulty of filtering. Increasing N lengthens the observation horizon and therefore increases the cumulative opportunity for weight degeneracy. The measles model is therefore a natural platform for testing whether guided intermediate resampling, and in particular neural guide construction, can improve filtering performance as the problem becomes more challenging.

4.3 Results Overview and Comparison Strategy

The results in this thesis are organized around a direct comparison of the three GIRF variants studied throughout the thesis, namely bootstrap-guide GIRF, moment-guide GIRF, and neural-guide GIRF. The central question is whether the neural guide improves practical filtering performance relative to the two classical guide constructions, while preserving the plug-and-play structure of the latent model and the exact observation-time likelihood evaluation. The comparison is therefore designed to assess not only which method gives the highest likelihood estimate on a fixed problem instance, but also how the three methods differ in numerical stability, computational cost, and failure mode as the problem dimension increases.

Several complementary perspectives are used in this comparison. The first concerns statistical performance, measured through normalized likelihood estimates and their variability

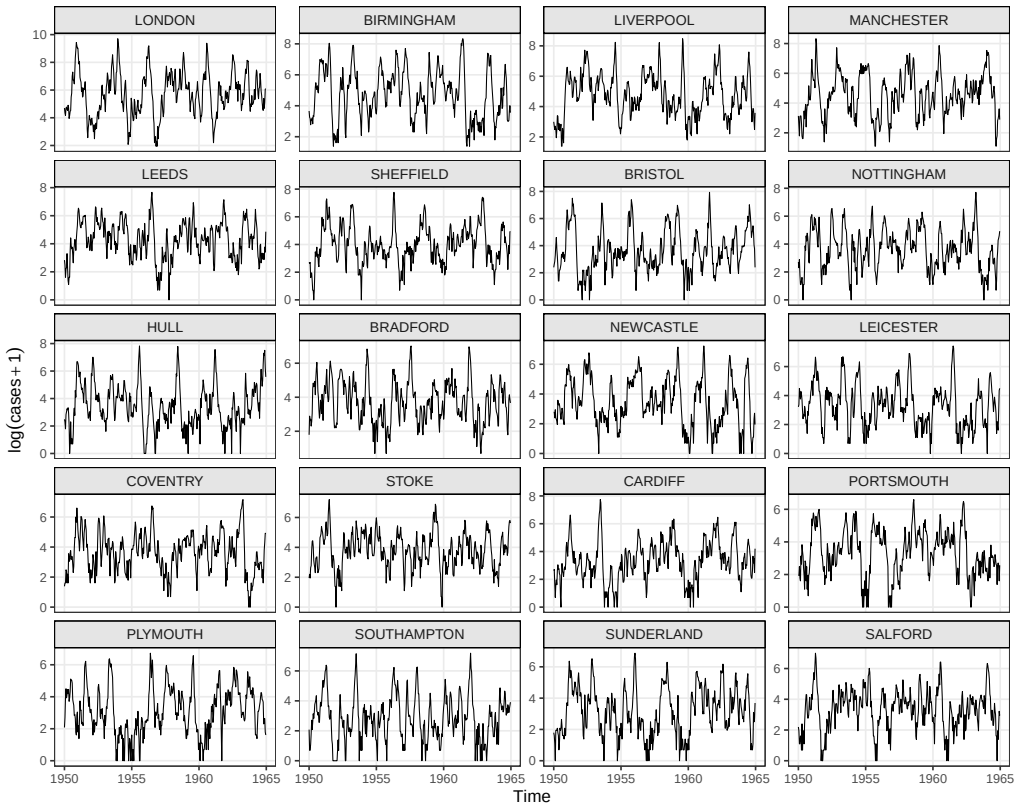


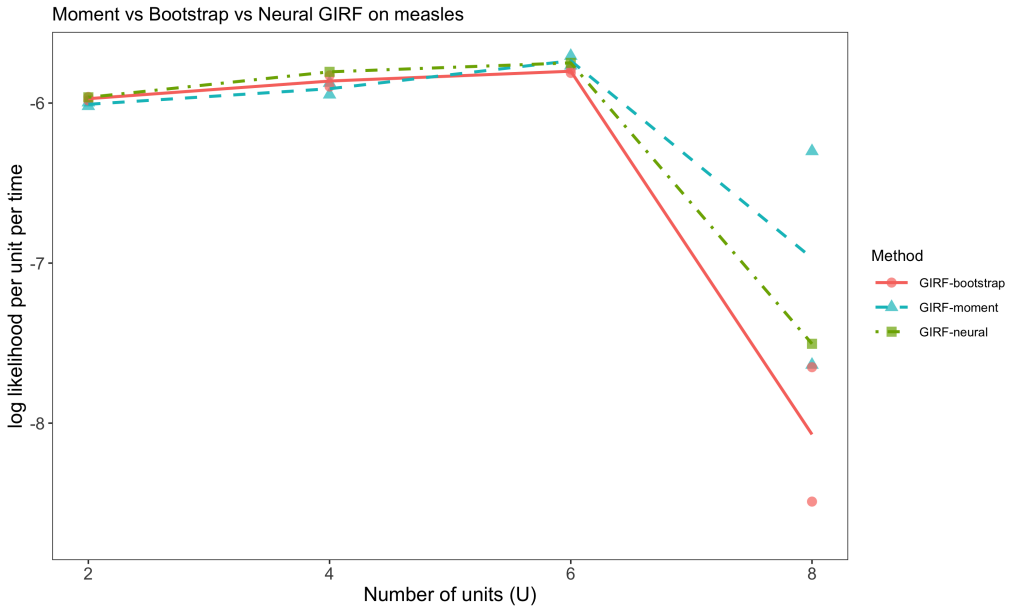
Figure 2: Illustration of measles case trajectories across cities in the SpatPOMP model. This figure provides a visual summary of temporal variability, between-city heterogeneity, and the structure of the observations that the filtering algorithms must explain.

across replicate runs. The second concerns computational scalability, measured through wall-clock runtime and, where feasible, peak memory usage. This is particularly relevant because simulation-heavy guide constructions may become impractical in higher-dimensional settings even before their statistical accuracy deteriorates. The third concerns diagnostic behavior within the time series itself. Since the measles model is heterogeneous across both cities and time, aggregate likelihood summaries alone do not reveal whether a method fails uniformly or only at a small number of particularly difficult intervals.

4.4 Three-Way Comparison of GIRF Variants

The main empirical comparison therefore places bootstrap-guide GIRF, moment-guide GIRF, and neural-guide GIRF on the same scale. Each method is evaluated on the same measles data, under matched latent dynamics, matched measurement model, and comparable filtering budgets. This three-way comparison is intended to reveal whether the neural guide behaves like a more accurate approximation to a classical guide, or whether it changes the qualitative scaling behavior of the filter as the number of units increases.

Figure 4.4 summarizes the main three-way comparison between bootstrap-guide GIRF,



moment-guide GIRF, and neural-guide GIRF on the measles model as the number of units increases. The horizontal axis varies the number of cities included in the filtering problem, while the vertical axis reports the estimated log likelihood normalized by the number of units and the number of observation times. This normalization makes it possible to compare behavior across systems of different effective dimension on a common scale.

Several features are immediately apparent. For small and moderate values of U, specifically $U = 2, 4$, and 6, all three methods perform similarly. The normalized log-likelihood values lie in a narrow range, and no method shows a decisive advantage. This suggests that in relatively low-dimensional settings the choice of guide construction is less important, at least for the present measles model and filtering budget. In this regime, the intermediate resampling structure itself appears to provide the main benefit, while the exact form of the guide only weakly affects the final likelihood estimate.

The picture changes markedly at $U = 8$. At this point the three methods separate, and the ranking becomes more informative about their robustness under increasing dimension. The bootstrap-guide GIRF shows the most severe degradation, with a substantial drop in normalized log likelihood relative to its performance at smaller values of U. The neural-guide GIRF also deteriorates, but to a noticeably smaller extent. In this sense, the neural guide provides a clear improvement over the bootstrap-style guide in the most difficult regime examined in this comparison. The moment-guide GIRF remains the strongest of the three at $U = 8$, achieving the least negative normalized log likelihood among the methods displayed in the figure.

This pattern suggests that the neural guide is capturing useful predictive structure, but not yet uniformly outperforming the best hand-designed guide. Up to $U = 6$, the neural method tracks the classical GIRF baselines closely, which is encouraging because it indicates that replacing the hand-constructed guide with a learned score does not degrade performance in easier regimes. At $U = 8$, the neural guide appears to preserve more stability than

the bootstrap guide, but it does not fully match the strongest performance obtained by the moment guide. A natural interpretation is that the learned guide already provides a nontrivial approximation to the predictive likelihood, but still leaves room for improvement in the most challenging part of the state space.

The spread of points at fixed U also indicates that Monte Carlo variability becomes more pronounced in the higher-dimensional setting. This is especially visible at $U = 8$, where the replicate-to-replicate differences are much larger than at $U = 2, 4$, or 6. Thus the main transition in Figure 4.4 is not only a decline in mean likelihood, but also an increase in numerical instability. Taken together, these results support the following conclusion. In the tested measles examples, all three GIRF variants are comparable in lower dimensions, but their differences become meaningful once the filtering problem becomes sufficiently difficult. In that regime, the neural guide improves substantially over the bootstrap-style guide, while the moment guide remains the strongest baseline among the methods considered here.

These findings motivate the subsequent diagnostic analyses, which examine whether the remaining gap between moment-guide GIRF and neural-guide GIRF can be attributed to a small number of difficult observation intervals rather than to uniform underperformance across the entire time series.

4.5 Diagnostic Comparisons by City and Time

Aggregate likelihood summaries are useful for comparing filtering methods, but they do not reveal whether poor performance is diffuse across the full dataset or concentrated in a small number of especially difficult observation intervals. To address this, diagnostic summaries were constructed at the interval level. For each observation time, the conditional log-likelihood contributions were summed over intermediate substeps, producing an interval-wise measure of filtering difficulty. This makes it possible to determine whether NN-GIRF fails gradually across the full time series or whether a few isolated intervals dominate the negative tail of the likelihood.

Figures 3 and 4 together show that the principal failure mode in this example is highly localized in time. Figure 4 shows that most observation intervals contribute at a relatively stable level, while only a very small number of intervals produce dramatically more negative likelihood contributions than the rest. This indicates that the degradation of filtering performance is not spread uniformly across the entire time series. Instead, a few difficult intervals dominate the likelihood loss.

Figure 3 helps interpret the source of these difficult intervals. The worst intervals occur at times when nearly all cities are undergoing rapid changes in epidemic intensity. In other words, the filtering difficulty is not driven by a single city behaving anomalously while the others remain stable. Rather, these problematic intervals correspond to system-wide transition periods in which the epidemic trajectories of many cities change sharply at the same time. During such periods, the future observation becomes especially sensitive to small errors in the intermediate latent state, and even a modest guide mismatch can be amplified through intermediate resampling.

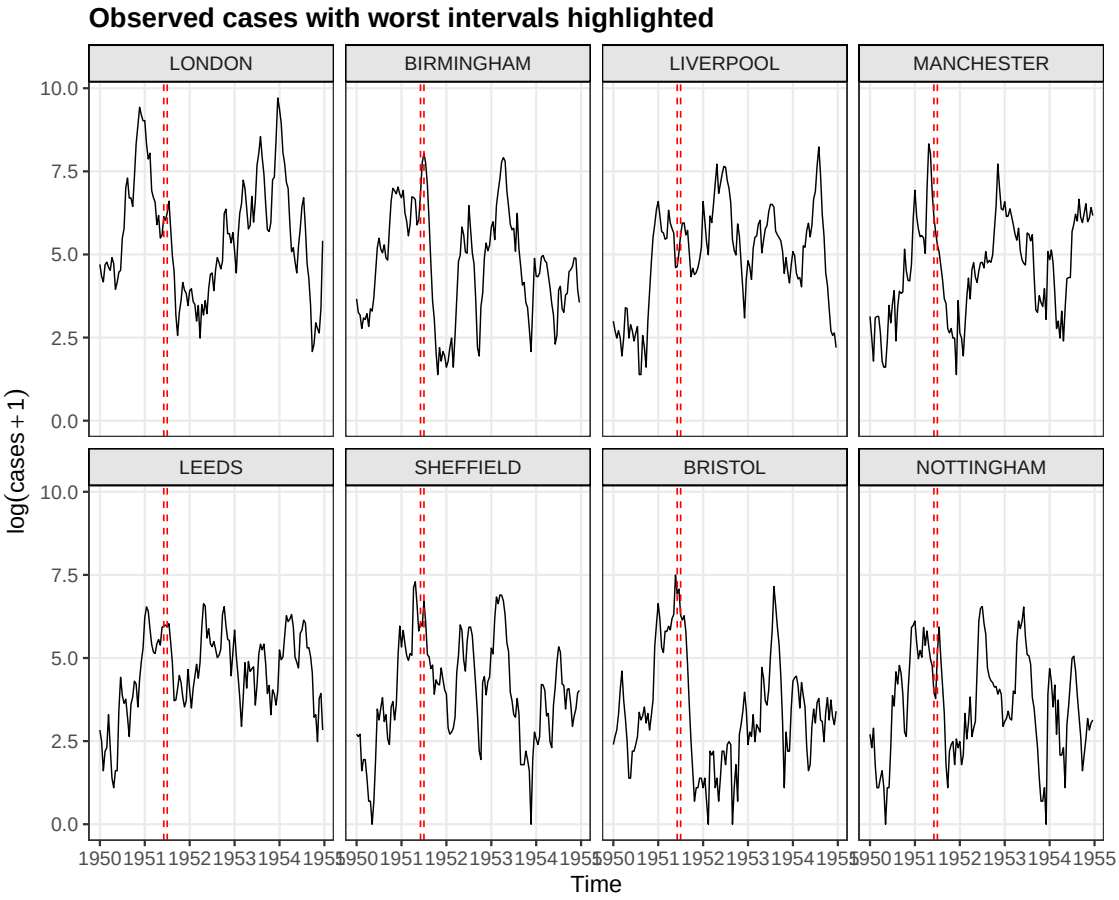


Figure 3: Observed biweekly measles case trajectories for the $U = 8$ example, shown on the $\log(\text{cases} + 1)$ scale. The dashed vertical lines mark the worst observation intervals identified by NN-GIRF. These intervals occur during periods in which many cities experience rapid changes in epidemic intensity simultaneously.

This pattern is informative for method development. If poor performance were distributed broadly over time, that would suggest a systematic mismatch between the learned guide and the latent process model. By contrast, the present diagnostics suggest that NN-GIRF performs reasonably well over most of the observation window and struggles primarily during a small number of abrupt transition periods. This points to a specific direction for improvement. Rather than only optimizing average guide quality over the whole training distribution, future guide construction should place additional emphasis on accurately representing intervals in which many units are simultaneously entering or exiting outbreak phases.

The current diagnostics therefore support two conclusions. First, the main source of instability in this example is temporal localization rather than a persistent failure across all intervals. Second, the most difficult intervals are associated with collective, system-wide changes in epidemic activity rather than isolated city-specific outliers. Both observations are useful for understanding the practical strengths and limitations of neural-guided intermediate resampling on the measles model.

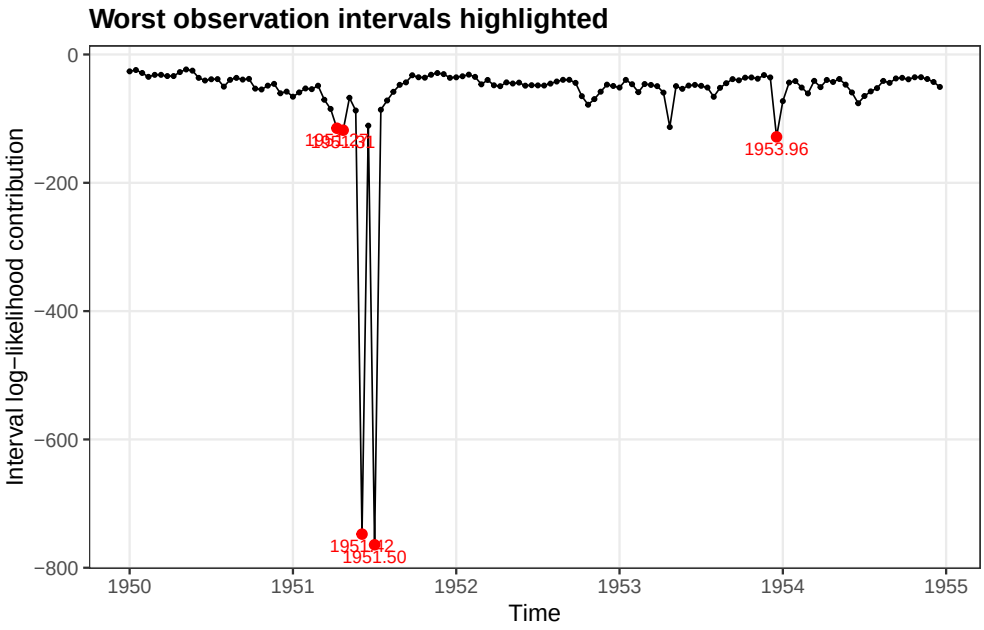


Figure 4: Interval-wise log-likelihood contribution for NN-GIRF in the $U = 8$ example. The highlighted points correspond to the worst observation intervals. Most intervals contribute at a relatively stable level, while a small number of intervals account for a disproportionate share of the total likelihood degradation.

5 Discussion

5.1 Interpreting the three GIRF variants in statistical and computational terms

The empirical results suggest that the main differences between bootstrap-guide GIRF, moment-guide GIRF, and neural-guide GIRF do not arise from the basic filtering skeleton, which is shared across all three methods, but from the way future observational information is approximated at intermediate times. This is consistent with the methodology developed earlier in the thesis. Once the intermediate resampling schedule is fixed, the dominant source of variation between GIRF variants is the quality and computational cost of the guide. It is therefore natural to interpret the three methods not as entirely different filters, but as three different approximations to the same ideal predictive guide.

Bootstrap-guide GIRF is the most direct construction conceptually. It attempts to approximate the predictive compatibility of future observations by repeated forward simulation. This makes it attractive when guide quality is judged purely in terms of fidelity to the latent simulator. However, that fidelity comes at a computational price. For each interval, particle, and intermediate step, the method may require forecast simulations or additional predictive evaluations whose cost grows with the number of particles, the number of intermediate steps, and the dimension of the latent state. In spatiotemporal models this also implies an increasing storage burden, because higher-dimensional forecast objects must be maintained long

enough to construct guide weights. Bootstrap-style guidance can therefore be viewed as statistically direct but computationally expensive. In higher-dimensional problems, the method may become difficult not only because the filtering task is hard, but also because the guide mechanism itself consumes substantial memory and time.

Moment-guide GIRF compresses future predictive information into a lower-dimensional approximation, typically through forecast means and variances (Park and Ionides, 2020). In terms of time and space complexity, this is often the most attractive of the three approaches. Once a suitable deterministic skeleton and variance construction are available, the guide can be evaluated using a small number of summary quantities rather than a large collection of guide simulations. This tends to reduce both runtime and memory usage. The price is approximation bias. In nonlinear, non-Gaussian, and discrete latent systems, low-order moment summaries may fail to preserve the shape of the predictive distribution that matters for resampling. The moment guide can therefore be interpreted as a low-variance but potentially misspecified approximation. The empirical results in this thesis suggest that, for the measles model and the dimensions tested here, this approximation remains remarkably competitive, and in some regimes stronger than the learned neural guide.

Neural-guide GIRF occupies an intermediate position. Like the moment guide, it avoids repeated online guide simulation. Like the bootstrap guide, it aims to approximate the predictive likelihood more flexibly rather than forcing that likelihood into a low-order parametric form. The computational burden is shifted from the online filtering phase to an offline amortized training phase. Once trained, the neural guide is evaluated by a forward pass of a learned score network, so the online cost scales primarily with the number of particles and the cost of the network architecture. In principle, this can produce a favorable time-memory trade-off relative to bootstrap-style guidance, especially when the same guide is reused across many filtering runs. At the same time, the quality of the neural guide depends critically on the relationship between the training distribution and the runtime distribution of intermediate states. The exact derivation in Section 3.1 makes this point explicit. The ideal guide is tied to the predictive intermediate-state distribution $\pi_{n,s}$, and a neural score trained on a mismatched state distribution can perform well on its training objective while still failing at the intervals that matter most during filtering.

From this perspective, the three GIRF variants can be understood as different ways of managing the same tension. Bootstrap guidance reduces model misspecification at the cost of substantial online computation and memory. Moment guidance reduces online cost by imposing a strong structural approximation on the predictive distribution. Neural guidance attempts to learn that predictive structure from simulation, thereby reducing online cost without fully committing to a hand-crafted low-order approximation. The results of this thesis suggest that this third strategy is viable, but not yet uniformly superior. In the present measles experiments, the neural guide substantially improves over the bootstrap-style guide in the most difficult tested regime, but does not consistently outperform the moment guide. This suggests that the dominant open problem is not whether neural guidance is useful in principle, but how to train and deploy it so that its flexibility is concentrated on the intervals where the classical approximations fail.

Although a full runtime and memory benchmark was not completed in the present study,

the algorithmic structures already suggest a qualitative ordering. Bootstrap-style guidance is expected to be the most expensive online because it relies on repeated forward predictive construction. Moment-guided GIRF is expected to be the lightest online because it collapses the guide to low-order summary quantities. Neural-guided GIRF lies between these extremes. Its online evaluation is cheaper than repeated guide simulation, but more expensive than a closed-form moment expression, while its offline training cost must also be taken into account. In applications where the guide is reused many times, for example repeated likelihood evaluations or parameter search, amortization makes the neural approach more attractive. In applications where only a small number of filter runs are required, a hand-crafted moment guide may remain preferable if it is already accurate enough.

5.2 Concluding remarks

This thesis set out to investigate whether neural posterior estimation ideas could be used to improve guided intermediate resampling in implicit spatiotemporal filtering problems. The answer provided by the present study is cautiously positive. On the theoretical side, the ideal GIRF guide was shown to admit an exact posterior-based reformulation, which provides a principled connection between guide construction and simulation-based learning. On the methodological side, this exact reformulation led naturally to a contrastive score-based implementation that can be inserted into the GIRF recursion without changing the plug-and-play structure of the latent model. On the empirical side, the resulting neural-guided GIRF remained competitive with classical GIRF baselines in lower-dimensional settings and improved substantially over the bootstrap-style guide in the most difficult case examined here.

At the same time, the results do not support the stronger claim that the neural guide is already the best available guide for the measles model. The moment guide remained highly competitive and, in some of the experiments presented here, stronger than the learned guide. The diagnostic analyses suggest that the remaining gap is not due to uniform underperformance across the entire time series, but is instead driven by a small number of especially difficult observation intervals associated with rapid, system-wide changes in epidemic intensity. This is an encouraging outcome, because it points to a specific failure mode rather than a general collapse of the neural approach. The main conclusion is therefore not that neural guidance has solved the guide-construction problem, but that it provides a viable and principled route toward more flexible guided filtering in implicit spatiotemporal models.

5.3 Future directions

One natural direction for future work is to make guide training more adaptive to the actual failure modes observed during filtering. The diagnostic results in this thesis suggest that poor performance is often concentrated in a small number of difficult observation intervals rather than spread uniformly across the entire time series. This suggests a targeted retraining strategy. One could first run NN-GIRF with an initial guide, identify the intervals with especially poor conditional log-likelihood contributions, and then generate additional training

data concentrated around those intervals or around similar latent regimes. The guide could then be fine-tuned on this targeted dataset. Such a procedure would preserve the amortized structure of the method while making the training distribution better aligned with the parts of the state space that dominate filtering failure. In effect, this would move the method from a purely offline guide to a partially adaptive guide that learns from its own errors.

A second direction concerns the expressive family used for guide approximation. The present implementation uses a contrastive score model, which is computationally convenient and sufficient for relative particle ranking. However, the exact derivation in Section 3.1 shows that guide construction can be framed through posterior estimation and posterior-to-prior ratios. This suggests that richer conditional density models may be useful, especially in higher-dimensional settings. Normalizing flows provide one such possibility. Flows were introduced as a flexible family of invertible density transformations by Rezende and Mohamed (2015), and have since developed into a mature framework for probabilistic modeling and inference (Rezende and Mohamed, 2015; Papamakarios et al., 2021). In the context of neural guidance, flow-based conditional models could be used either to approximate the intermediate-state posterior more directly or to learn a more structured approximation to the predictive observation distribution. Whether this additional flexibility improves filtering performance enough to offset the increased model complexity remains an open empirical question.

A related statistical direction is to tighten the link between training and filtering. In the present thesis, the guide is trained from simulated pairs that approximate the runtime law of intermediate states and future observations. This is already an improvement over training only on observation-time states, but it remains an offline approximation. A stronger alternative would be to generate training data from the states actually visited by a running filter and then fine-tune the guide on those states. Such an iterative procedure would more closely match the posterior-to-prior ratio that appears in the exact reformulation of the ideal guide. It would also create a natural bridge between amortized simulation-based inference and online filtering diagnostics.

Finally, there is a clear software direction for making neural-guided filtering easier to use. At present, the implementation developed in this thesis relies on the mature `spatPomp` package in R for the filtering loop, while neural training and guide evaluation are carried out in Python. This split was practical for research development, since `spatPomp` provides a stable and general implementation of SpatPOMP models and GIRF (Asfaw et al., 2024). However, the split environment also introduces friction. Objects must be moved across language boundaries, threading and runtime environments must be carefully controlled, and debugging becomes more difficult when filtering and learning live in different ecosystems. A Python-native workflow would therefore be attractive. The current `pypomp` project provides a Python framework for POMP modeling and inference, with tutorials, quantitative tests, and documentation, while remaining under active and still early-stage development (Abkemeier et al., 2024). A unified Python implementation in which simulation, neural training, and filtering are all carried out in the same environment may substantially simplify future work on neural-guided particle methods.

References

Abkemeier, A., Chen, J., Ionides, E., Wheeler, J., and Tan, K. (2024). pypomp. https://github.com/pypomp/pypomp.

Asfaw, K., Park, J., King, A. A., and Ionides, E. L. (2021). A tutorial on spatiotemporal partially observed Markov process models via the R package spatpomp. *arXiv preprint arXiv:2101.01157*.

Asfaw, K., Park, J., King, A. A., and Ionides, E. L. (2024). spatPomp: An R package for spatiotemporal partially observed Markov process models. *Journal of Open Source Software*, 9(104):7008.

Bengtsson, T., Bickel, P., and Li, B. (2008). Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In *Probability and statistics: Essays in honor of David A. Freedman*, volume 2, pages 316–335. Institute of Mathematical Statistics.

Brehmer, J. and Cranmer, K. (2022). Simulation-based inference methods for particle physics. In *Artificial Intelligence for High Energy Physics*, pages 579–611. World Scientific.

Bretó, C., He, D., Ionides, E. L., and King, A. A. (2009). Time series analysis via mechanistic models. *The Annals of Applied Statistics*, pages 319–348.

Chatha, P., Bu, F., Regier, J., Snitkin, E., and Zelner, J. (2024). Neural posterior estimation for stochastic epidemic modeling. *arXiv preprint arXiv:2412.12967*.

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

Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer.

Evensen, G. (2003). The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *International conference on machine learning*, pages 2404–2414. PMLR.

He, D., Ionides, E. L., and King, A. A. (2010). Plug-and-play inference for disease dynamics: measles in large and small populations as a case study. *Journal of the Royal Society Interface*, 7(43):271–283.

Hermans, J., Begy, V., and Louppe, G. (2020). Likelihood-free MCMC with amortized approximate ratio estimators. In *International conference on machine learning*, pages 4239–4248. PMLR.

Ionides, E. L., Asfaw, K., Park, J., and King, A. A. (2023). Bagged filters for partially observed interacting systems. *Journal of the American Statistical Association*, 118(542):1078–1089.

King, A. A., Nguyen, D., and Ionides, E. L. (2016). Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, 69:1–43.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and Bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288.

Moral, P. (2004). *Feynman-Kac formulae: genealogical and interacting particle systems with applications*. Springer.

Naesseth, C., Linderman, S., Ranganath, R., and Blei, D. (2018). Variational sequential monte carlo. In *International conference on artificial intelligence and statistics*, pages 968–977. PMLR.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Papamakarios, G. and Murray, I. (2016). Fast ε-free inference of simulation models with bayesian conditional density estimation. *Advances in neural information processing systems*, 29.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Papamakarios, G., Sterratt, D., and Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pages 837–848. PMLR.

Park, J. and Ionides, E. L. (2020). Inference on high-dimensional implicit dynamic models using a guided intermediate resampling filter. *Statistics and Computing*, 30(5):1497–1522.

Pitt, M. K. and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599.

Poterjoy, J. (2016). A localized particle filter for high-dimensional nonlinear systems. *Monthly Weather Review*, 144(1):59–76.

Rebeschini, P. and Van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR.

Snyder, C., Bengtsson, T., Bickel, P., and Anderson, J. (2008). Obstacles to high-dimensional particle filtering. *Monthly Weather Review*, 136(12):4629–4640.

A Appendix: Implementation Notes

The implementation used in this thesis is available in the project repository `https://github.com/Shugeoy/Neural-GIRF.git`. This appendix records the practical workflow used to generate datasets, train neural guides, and run filtering experiments. The intention is to make the project reproducible for future readers, especially readers who are familiar with the statistical methodology but not yet with the software pipeline.

A.1 Code organization

The implementation consists of three main components. The first component is the spatiotemporal model specification and filtering code in R, built on top of `pomp` and `spatPomp`. The second component is the neural guide training code in Python, which is used to learn a simulation-based approximation to the intermediate guide. The third component is the runtime bridge between the two languages, implemented through `reticulate`, so that a guide trained in Python can be queried from within the R filtering loop.

In the version used for this thesis, the core files were:

```
measlesModel/measles_spatPomp.rda
nngirf.R
nngirf_diag.R
nnguide_runtime.py
trainTF_intermediate_csv.py
```

The file `measles_spatPomp.rda` stores the simulated measles object and the helper function `measles_subset()`, which allows a common model object to be restricted to smaller numbers of cities and observation times. The file `nngirf.R` implements the neural-guided filtering loop. The file `nngirf_diag.R` provides a diagnostic version of the same algorithm. The file `trainTF_intermediate_csv.py` trains the neural guide from intermediate-state simulation data. The file `nnguide_runtime.py` loads the trained neural model and exposes a guide-evaluation function to R.

A.2 Software environment

The filtering code was run in R and the guide model was trained in Python. In the implementation used in this thesis, the Python environment was accessed from R through `reticulate`. A practical difficulty encountered during development was that simultaneous use of R, `reticulate`, NumPy, and PyTorch could lead to low-level threading conflicts on macOS. In practice, stable execution required fixing the Python interpreter explicitly and constraining some thread settings before loading `reticulate`. The following initialization pattern was used at the beginning of interactive R sessions:

```
Sys.setenv(
```

```
  RETICULATE_PYTHON = "/Users/shugeoy/.virtualenvs/nngirf/bin/python",
  MKL_NUM_THREADS = "1",
  MKL_DYNAMIC = "FALSE",
  OMP_NUM_THREADS = "1",
  OMP_DYNAMIC = "FALSE"
)

library(reticulate)
py_run_string("
import torch
torch.set_num_threads(1)
torch.set_num_interop_threads(1)
")
```

The precise values of these threading variables may differ across machines, but recording them is important because filtering performance and stability were sensitive to low-level runtime settings.

The R packages most frequently used in this project were

```
pomp
spatPomp
ggplot2
dplyr
tidyr
patchwork
reticulate
```

and the Python side relied primarily on

```
numpy
pandas
torch
```

A virtual environment can be created and populated using

```
python3 -m venv ~/.virtualenvs/nngirf
source ~/.virtualenvs/nngirf/bin/activate
pip install --upgrade pip setuptools wheel
pip install numpy pandas torch
```

The activation step above is a shell command and should be run in the terminal, not in the R console.

A.3 Generating the base measles object

The starting point for all experiments is the file `measlesModel/measles_spatPomp.rda`. This file stores a large simulated measles object together with the helper function `measles_subset()`, which constructs a smaller benchmark problem by selecting the first U cities and the first N observation times. Once the file has been created, the typical workflow in R begins with

```
load("measlesModel/measles_spatPomp.rda")
```

After loading, a concrete test object is created by

```
N_obs <- 5 * 26
U <- 8
m_obj <- measles_subset(m_U = U, m_N = N_obs)
```

This creates a five-year, eight-city filtering problem. In the experiments, the same stored simulation object was used to construct multiple benchmark problems with different values of U and N, ensuring that the latent model and parameterization remained fixed while the effective dimension changed.

A.4 Generating intermediate-state training data

The neural guide was not trained on observation-time states alone. Instead, the training data were generated by recording intermediate latent states inside each observation interval and pairing them with the future observation at the end of that interval. This was done because the guide is evaluated at intermediate times during filtering, and therefore the training distribution should resemble the runtime distribution of intermediate states as closely as possible.

A typical data-generation call in R had the form

```
train_csv_path <- simulate_measles_train_csv_intermediate(
  U = 20,
  years = 15,
  n_sims = 200,
  seed = 2025,
  prefix = "measles20_ns200",
  Ninter = 20,
  overwrite = TRUE
)
```

The resulting CSV file contains one row per unit and per retained intermediate state. In the implementation used in this thesis, the stored fields included identifiers for the simulation, interval, and substep, the intermediate time, the future observation time, the city label,

the future reported cases, and the intermediate latent states needed as neural inputs. The exact long-format schema is implementation-specific and therefore omitted from the main methodology chapter, but the key conceptual point is that the training examples correspond to pairs

$$(\mathbf{X}_{n,s}, \mathbf{Y}_n)$$

for $s = 1, \ldots, S-1$, rather than only $(\mathbf{X}_{n-1}, \mathbf{Y}_n)$.

In higher-dimensional settings, storing every substep can become expensive. A more memory-efficient alternative is to keep only a subset of representative substeps from each interval. This reduces the size of the training set while still covering the range of intermediate-state distributions seen during filtering.

A.5 Neural guide architecture

The neural guide used in the main experiments was implemented as a direct score model. Its purpose is not to estimate a normalized posterior density explicitly, but to produce a scalar score

$$s_\phi(\mathbf{x}, \mathbf{y})$$

that ranks intermediate states according to their compatibility with the future observation block. In the main experiments $L = 1$, so the future block reduces to the next observation $\mathbf{y}_n$.

The input state tensor has shape

$$(B, U, F_X),$$

where B is the batch size, U is the number of cities, and F_X is the number of state-side features per city. In the implementation used in the thesis,

$$F_X = 7,$$

with features

$$\left[\frac{S}{P},\ \frac{E}{P},\ \frac{I}{P},\ \frac{R}{P},\ \log P,\ \sin(2\pi t_x),\ \cos(2\pi t_x)\right],$$

where $P = S + E + I + R$ is the city population represented in the latent state and t_x is the intermediate time. The observation-side tensor has shape

$$(B, U, F_Y),$$

with

$$F_Y = 3,$$

and features

$$[\log(1 + \text{cases}),\ \sin(2\pi t_y),\ \cos(2\pi t_y)],$$

where t_y is the future observation time.

Each city is first encoded by a small multilayer perceptron. Separate encoders are used for the state and observation branches. The resulting city-wise embeddings are then passed through

transformer encoder layers across the unit dimension. This allows the model to represent cross-city interactions while preserving a shared per-city representation. After transformer encoding, embeddings are pooled across cities and projected into a common score space. The final similarity score is

$$s_\phi(\mathbf{x}, \mathbf{y}) = \frac{\langle z_\phi^X(\mathbf{x}), z_\phi^Y(\mathbf{y}) \rangle}{\tau},$$

where z_ϕ^X and z_ϕ^Y are normalized embeddings of the state and future observation, and τ is a temperature parameter. In the main implementation, the transformer width was

$$d = 64,$$

with two transformer layers and eight attention heads in the larger experiments.

A.6 Training objective and batching strategy

The neural score was trained with the InfoNCE objective. For a batch of positive pairs

$$(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(B)}, \mathbf{y}^{(B)}),$$

define the score matrix

$$S_{ij} = s_\phi(\mathbf{x}^{(i)}, \mathbf{y}^{(j)}).$$

The loss is

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}.$$

This trains the network to assign higher scores to matched pairs than to mismatched pairs. The corresponding score behaves like a conditional density ratio and is therefore suitable for ranking particles during intermediate resampling.

An important practical detail is the batching strategy. In the implementation used in this thesis, examples were grouped by a common `step_id`. This means that all examples within a batch correspond to the same relative intermediate position within the observation interval. This design prevents the network from solving the task primarily through time features, and instead forces it to use the latent state information to distinguish which future observation belongs with which intermediate state.

A.7 Training commands

Once the intermediate-state dataset had been generated, the neural guide was trained in Python. A typical command for the five-city training set was

```
python trainTF_intermediate_csv.py \
  --csv measles5_U5_T15yrs_SEIR_C_interNinter5.csv \
  --out guide_U5.pt \
  --epochs 50 \
```

```
  --batch-size 160 \
  --device cpu \
  --n-heads 8 \
  --seed 0 \
  --split-seed 0
```

For the $U = 20$ case, one representative training run was

```
python trainTF_intermediate_csv.py \
  --csv measles20_ns200_U20_T15yrs_SEIR_C_interNinter20.csv \
  --out guide_U20_full19.pt \
  --epochs 12 \
  --batch-size 160 \
  --device cpu \
  --n-heads 8 \
  --seed 0 \
  --split-seed 0
```

The smaller number of epochs in the larger experiment was used because each epoch already contains many more optimization steps when the number of retained substeps is large.

In the implementation used for the thesis, the training script automatically saved the best checkpoint according to validation loss. This was important because the neural guide could over-sharpen as training continued, even when the contrastive loss continued to improve.

A.8 Running NN-GIRF in R

After training, the guide is loaded into R through `reticulate` and used inside the filtering loop. A standard evaluation workflow was

```
Sys.setenv(
  RETICULATE_PYTHON = "/Users/shugeoy/.virtualenvs/nngirf/bin/python",
  MKL_NUM_THREADS = "1",
  MKL_DYNAMIC = "FALSE",
  OMP_NUM_THREADS = "1",
  OMP_DYNAMIC = "FALSE"
)

library(reticulate)
py_run_string("
import torch
torch.set_num_threads(1)
torch.set_num_interop_threads(1)
")
```

```
source("nngirf.R")
nngirf_setup_py("nnguide_runtime.py", "guide_U20_full19.pt", device = "cpu")

load("measlesModel/measles_spatPomp.rda")
N_obs <- 15 * 26
U <- 20
m_obj <- measles_subset(m_U = U, m_N = N_obs)
Ninter_val <- U
tol <- 1e-300

set.seed(123)
g_nn <- nngirf(
  m_obj,
  Np = 5000,
  Ninter = Ninter_val,
  lookahead = 1,
  params = coef(m_obj),
  tol = tol
)

logLik(g_nn) / (N_obs * U)
```

The final line computes the normalized log likelihood that was used throughout the empirical comparisons.

A.9 Running diagnostic versions and producing figures

For detailed diagnosis, a second implementation called `nngirf_diag()` was used. This version records progress information during filtering and stores interval-level diagnostic quantities such as substep-wise conditional log-likelihood contributions. A typical diagnostic run was

```
source("nngirf_diag.R")

set.seed(123)
g_nn_diag <- nngirf_diag(
  m_obj,
  Np = 5000,
  Ninter = Ninter_val,
  lookahead = 1,
  params = coef(m_obj),
  tol = tol,
  progress = TRUE,
```

```
  progress_every = 10,
  diag_save_path = "diag_U20_seed123.rds"
)
```

The saved diagnostic object can then be used to construct interval-level or city-by-time figures. In the $U = 8$ example discussed in the thesis, two diagnostic plots were particularly useful. The first displayed the observed case trajectories for all cities with the worst intervals highlighted. The second displayed the interval-wise log-likelihood contribution and identified the few intervals that dominated the negative tail of the total likelihood. This made it possible to determine whether poor performance was diffuse or concentrated in a small number of especially difficult periods.

A.10 Reproducibility notes

All reported experiments were run with fixed random seeds and a cached workflow. Long-running intermediate results were saved whenever possible so that experiments could be resumed without rerunning earlier stages. In practice, reproducibility required keeping together the following items:

1. the trained neural checkpoint,
2. the exact training CSV used to produce that checkpoint,
3. the `measles_spatPomp.rda` file used to construct benchmark objects,
4. the R and Python environment settings,
5. the filtering seed used for each evaluation run.

Because cross-language execution through `reticulate` was sensitive to environment configuration, recording software versions and thread settings was as important as recording model parameters and particle counts.