An Overview on Tracking Problems

Mohit Prashant
mohit010@e.ntu.edu.sg

Nanyang Technological University

Abstract

Modern financial systems generate vast quantities of sequential data characterized by non-linear dynamics and non-Gaussian noise distributions. Accurately modelling the latent states of these systems is necessary for risk management and decision making. This report explores time-series inference methods ranging from Kalman Filters (KF) for linear-Gaussian systems to advanced Particle Flow Filters (PFF) in non-linear scenarios. We conclude with an overview on differentiable filters and their application to tracking non-linear systems at risk of the contagion effect.

Keywords: Filtering, Particle Flow, State-Space Modelling

Contents

0.1 Introduction

Modern banking and financial systems generate vast streams of sequential data. Extracting actionable information from these data requires accurate inference of latent system states that evolve over time under uncertainty. These latent dynamics play a central role in applications ranging from risk management and portfolio optimization to anomaly detection and monitoring. State Space Models (SSMs) provide a framework for modelling these systems by pairing hidden state evolution with noisy [1, 2].

Non-Linear Dynamics

For linear-Gaussian SSMs, the Kalman Filter (KF) offers an exact and computationally efficient solution. However, financial systems are rarely linear or Gaussian in practice; they exhibit strong non-linearities, heavy-tailed noise and high-dimensional dependencies. In these settings, classical extensions such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) often suffer from linearization error, numerical instability, or breakdown under strong non-linearity [2]. Particle Filters (PFs), based on sequential Monte Carlo methods, provide an alternative capable of handling non-linear models. However, they suffer from particle degeneracy issues and tend to scale poorly to high dimensional spaces [3].

Particle Flow Filters

Particle Flow Filters (PFFs) have emerged as a class of methods designed to address these limitations by deterministically or stochastically transporting particles from prior to posterior distributions without explicit resampling. By constructing continuous-time flows that approximate Bayesian updates, particle flow methods aim to mitigate degeneracy while improving numerical stability and efficiency [4]. More recent developments extend the PFF framework by introducing stochastic flows for stiffness mitigation and by making particle filters differentiable end-to-end, enabling integration with modern deep learning pipelines [5].

Report Overview

We begin by revisiting linear and non-linear filtering in SSMs to establish baseline performances and limitations [1, 6–8]. We then study deterministic, kernel-based and stochastic particle flow filters, analyzing their stability, scalability and empirical behavior across more complex SSMs [9, 10]. Building on this, we design and evaluate differentiable particle filters using entropy-regularized optimal transport resampling, enabling backpropagation through sequential inference [11]. Finally, we investigate extensions to gradient-based Bayesian inference via neural acceleration of optimal transport and we examine SSMs with more complex causal relationships between states, i.e. contagion effects [12].

0.2 Literature Review

0.2.1 Classical Foundations and the Degeneracy Problem

The Sequential Importance Resampling (SIR) filter, popularized in [13], remains the baseline for non-linear SSMs. However, its reliance on weight-based discrete sampling leads to particle degeneracy phenomenons. In high-dimensional financial manifolds, the variance of importance weights grows exponentially with the state dimension, causing the effective sample size to collapse [3]. While the Unscented Kalman Filter (UKF) addresses non-linearity via deterministic sigma-point sampling, it remains constrained by Gaussian assumptions that fail to capture the multi-modality of market crash events [8, 14].

0.2.2 Particle Flow: Transport-Based Inference

To bypass the limitations of importance sampling, [10] proposed the *Particle Flow Filter* (PFF). Rather than weighting, PFFs utilize a vector field to migrate particles from the prior to the posterior.

- **Exact and Local Flows:** The Exact Daum-Huang (EDH) flow computes a deterministic path using the gradient of the log-likelihood. [9] refined this with the Invertible Particle Flow (PF-PF), ensuring the flow acts as a diffeomorphism, which is critical for maintaining the probabilistic integrity of the particles.
- **Kernel-Embedded Methods:** [3] introduced RKHS-based flows to handle high-dimensional system applications. By using matrix-valued kernels, these filters prevent the collapse of observed-variable marginals, a common failure point in sparse financial data.
- **Stochastic Extensions:** [5, 15] introduced stochasticity into the flow to mitigate stiffness: a numerical instability where the flow becomes overly sensitive to initial conditions. This stochasticity ensures better exploration of the state space.

0.2.3 Differentiable Particle Filters and Optimal Transport

The integration of PFs into gradient-based learning requires a differentiable resampling step. [11] solve this using *Entropy-Regularized Optimal Transport* (OT). By applying the Sinkhorn algorithm, the resampling process becomes a smooth mapping, allowing for the use of Hamiltonian Monte Carlo (HMC). As noted in [16], while Particle Marginal Metropolis-Hastings (PMMH) is a robust baseline, HMC-based parameter inference offers superior convergence rates in complex, high-dimensional parameter spaces.

0.2.4 Alternatives and Future Extensions

Beyond the scope of flow-based methods, several alternative paradigms offer distinct advantages:

Variational Sequential Monte Carlo (VSMC)

Proposed by [17], VSMC reinterprets the filtering problem as a variational optimization task. Instead of an analytical flow, it learns a proposal distribution by maximizing the Evidence Lower Bound. While often faster than Sinkhorn-based OT, it may introduce an approximation bias that PFFs theoretically avoid.

Gaussian Sum and Multiple Model Filters

For systems exhibiting regime-switching, Gaussian Sum Filters approximate the posterior as a mixture of Gaussians. Unlike PFs, these maintain analytical tractability for each component, making them highly efficient for low-dimensional systems with clear multi-modal structures [18].

Neural Operators and Amortized OT

The computational bottleneck of OT has led to Neural Acceleration. [19] introduced GradNetOT to learn the transport map directly, while [20] explored Neural Operators for scientific computing [21]. These methods seek to amortize the cost of inference, allowing a pre-trained network to predict the particle flow in a single forward pass—a crucial requirement for low-latency trading environments.

Ensemble Kalman Filters (EnKF)

In extremely high-dimensional settings where computing Jacobians is infeasible, the Ensemble Kalman Filter [22] remains a formidable alternative. While technically biased for non-Gaussian noise, its numerical stability often outperforms PFs in geophysical and large-scale portfolio applications.

Chapter 1

Part 1 - From classical filters to particle flows

1.1 Filtering Under Linear Dynamics

This section serves as an introduction to Linear Gaussian State Space Models (LGSSM) as well as Kalman filters. The objective of filtering is to estimate the true system state given a noisy observation and some approximation of the system dynamics. Following Example 2 in [1], the Linear Gaussian model is defined on the state space $\mathcal{X} = \mathbb{R}^{n_x}$ and observation space $\mathcal{Y} = \mathbb{R}^{n_y}$.

1.1.1 LGSSM System Equations

The latent state process $\{X_n\}_{n\geq 1}$ and the observation process $\{Y_n\}_{n\geq 1}$ evolve according to the following linear stochastic difference equations:

$$X_n = AX_{n-1} + BV_n \tag{1.1}$$
$$Y_n = CX_n + DW_n \tag{1.2}$$

where:

- A, B, C, D are matrices of appropriate dimensions.
- $\{V_n\}$ and $\{W_n\}$ are independent and identically distributed (i.i.d.) Gaussian noise vectors.

1.1.2 Kalman Filtering

The Kalman filter provides the optimal recursive solution for the LGSSM by computing the posterior distribution $p(X_n|Y_{1:n}) = \mathcal{N}(\hat{X}_{n|n}, P_{n|n})$ [1]. Following the standard Bayesian recursion, the filter alternates between prediction and correction [6].

Prediction Step The filter projects the previous state estimate and covariance forward in time:

$$\hat{X}_{n|n-1} = A\hat{X}_{n-1|n-1} \tag{1.3}$$
$$P_{n|n-1} = AP_{n-1|n-1}A^\top + BB^\top \tag{1.4}$$

where $BB^\top$ represents the process noise covariance Q.

Update Step Given a new observation Y_n, the prediction is refined using the Kalman gain K_n:

$$\text{Innovation: } \tilde{Y}_n = Y_n - C\hat{X}_{n|n-1} \tag{1.5}$$
$$\text{Innovation Covariance: } S_n = CP_{n|n-1}C^\top + DD^\top \tag{1.6}$$
$$\text{Optimal Kalman Gain: } K_n = P_{n|n-1}C^\top S_n^{-1} \tag{1.7}$$
$$\text{Updated State: } \hat{X}_{n|n} = \hat{X}_{n|n-1} + K_n\tilde{Y}_n \tag{1.8}$$

Joseph Stabilized Covariance Updates

In practical implementations, the standard covariance update $P_{n|n} = (I - K_n C)P_{n|n-1}$ is prone to numerical instability. Round-off errors can lead to a loss of symmetry or, more critically, result in a matrix that is no longer positive-definite. To ensure numerical robustness, we utilize the Joseph stabilized form:

$$P_{n|n} = (I - K_n C)P_{n|n-1}(I - K_n C)^\top + K_n(DD^\top)K_n^\top \tag{1.9}$$

This formulation is computationally more expensive but guarantees that $P_{n|n}$ remains symmetric and positive semi-definite, provided $P_{n|n-1}$ and the measurement noise covariance $DD^\top$ are well-conditioned.

1.1.3 Experiments

We implement the LGSSM using synthetic data from a standard trajectory. We analyze the filter's performance by monitoring the conditioning number of the covariance matrices to detect potential divergence.

Relevant Metrics

To evaluate the filtering performance and numerical stability of the LGSSM, we utilize the following metrics:

- **Root Mean Squared Error (RMSE):** Measured as the sqrt of $\frac{1}{T}\sum_{n=1}^{T} ||X_n - \hat{X}_{n|n}||^2$ to quantify tracking accuracy.
- **Condition Number (κ):** We track $\kappa(P_{n|n}) = \frac{\lambda_{max}}{\lambda_{min}}$ to assess the numerical stability and risks of matrix inversion singularity in high-dimensional states.

Results

For posterity, we compare the Kalman filter with a simple mean filter on the LGSSM. The results table is as follows:

Table 1.1: Comparison of Filtering Performance on Synthetic LGSSM Data

Filter Type	**RMSE**
KF (Joseph Stabilized)	0.2277
KF (Regular)	0.2277
Mean Filter	1.3037

Numerical Stability Analysis The comparison between the standard Kalman Filter and the Joseph stabilized form reveals a high degree of consistency in state estimation, as evidenced by the identical RMSE values. However, differences in the evolution of the covariance matrices seem to arise due to floating-point arithmetic. The calculated max covariance difference ($\|P_{Joseph} - P_{Regular}\|_\infty$) is:

$$\Delta P_{max} = 2.21 \times 10^{-03}$$

While the state means remain aligned, this discrepancy highlights the effect of the Joseph form in re-symmetrizing the covariance update, which is critical for preventing divergence in long-duration financial time-series. We examine this in further detail in the following graph.

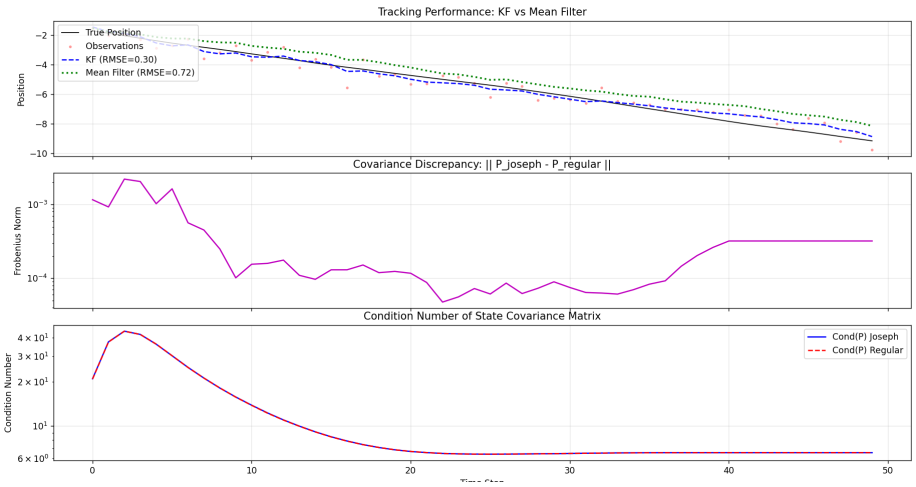

Figure 1.1: KF Tracking results showing ground truth vs. estimated state

1.2 Filtering Under Non-Linear Dynamics

Most real-world financial systems are described using non-linear dynamics, i.e. stochastic, heavy-tailed, non-Gaussian etc. As described in Example 4 of [1], the stochastic volatility state space model (SVSSM) is defined on the spaces $\mathcal{X} = \mathcal{Y} = \mathbb{R}$. Within this section, we apply modifications to the Kalman filtering method, i.e. EKF and UKF and use them to track the SVSSM alongside particle filtering (PF) methods.

1.2.1 SVSSM System Equations

The latent volatility process $\{X_n\}$ and the observation process $\{Y_n\}$ are defined as:

$$X_n = \alpha X_{n-1} + \sigma V_n \tag{1.10}$$
$$Y_n = \beta \exp(X_n/2) W_n \tag{1.11}$$

where:

- α is the persistence of the volatility.
- σ is the standard deviation of the process noise.
- β is a constant scaling factor for the observations.
- $V_n \sim \mathcal{N}(0, 1)$ and $W_n \sim \mathcal{N}(0, 1)$ are independent Gaussian noise terms.

To note, the model's density functions are given by:

- **Initial Distribution:** $X_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\alpha^2}\right)$, marginal distribution of X_n is stationary.
- **Transition Density:** $f(x'|x) = \mathcal{N}(x'; \alpha x, \sigma^2)$.
- **Observation Density:** $g(y|x) = \mathcal{N}(y; 0, \beta^2 \exp(x))$.

1.2.2 Non-Linearity and Non-Gaussianity

While the transition equation is linear in X_n, the observation equation is highly non-linear due to the exponential term $\exp(X_n/2)$. Furthermore, the marginal likelihood $p(y_n|x_n)$ is non-Gaussian, as it essentially represents a scale-mixture of Gaussians, a characteristic often found in high-frequency financial time series. Within the final section of this report we use the multivariate SVSSM to better emulate system dynamics.

1.2.3 Extended/Unscented Kalman Filters

To approximate the filtering distribution in the non-linear SVSSM, we first consider deterministic approximations based on the Kalman framework.

Extended Kalman Filter (EKF) The EKF linearizes the non-linear observation function $Y = \beta \exp(X/2)$ via a first-order Taylor expansion around the predicted state $\hat{x}_{n|n-1}$. The Jacobian H_n is computed as:

$$H_n = \frac{\partial h(x)}{\partial x}\Big|_{x=\hat{x}_{n|n-1}} = \frac{\beta}{2} \exp\left(\frac{\hat{x}_{n|n-1}}{2}\right) \tag{1.12}$$

In the SVSSM context, the EKF often provides poor estimates because the high curvature of the exponential mapping leads to significant linearization errors, particularly in high-volatility regimes [7].

Unscented Kalman Filter (UKF) The UKF utilizes the Unscented Transform to propagate a set of deterministically chosen sigma points through the non-linear observation equation. By capturing the mean and covariance up to the second order of the Taylor expansion, it typically outperforms the EKF. However, as the UKF still assumes a Gaussian posterior, it may fail to capture the skewness or multimodality that may arise in non-Gaussian financial dynamics [8].

Note on UKF/EKF Since $E[Y_n|X_n] = 0$, direct linearization fails. We apply the transformation $Z_n = Y_n^2$ and $Y = \beta^2 \exp(X/2)$ for the EKF and UKF measurement updates. Within our work, we use TF.Autograd to approximate the Jacobian instead of explicitly calculating it using the aforementioned Taylor Expansion analytically.

1.2.4 Particle Filters

The PF provides a Sequential Monte Carlo (SMC) approximation of the filtering distribution $p(x_n|y_{1:n})$ using a weighted set of N particles $\{W_n^i, X_n^i\}_{i=1}^N$.

Sequential Importance Resampling (SIR) A limitation we faced in our implementation was rapid particle degeneracy. To combat this, we applied a resampling method within the particle filter that takes effect when the degeneracy crosses a predefined threshold that serves as a hyperparameter for our algorithm [13]. We implement the SIR algorithm where the transition prior $f(x_n|x_{n-1})$ is used as the proposal density. The incremental weights are calculated as:

$$\alpha_n(X_{n-1:n}^i) = g(y_n|X_n^i) = \mathcal{N}(y_n; 0, \beta^2 \exp(X_n^i)) \tag{1.13}$$

This approach allows the filter to adapt to arbitrary non-linearities without requiring functional approximations.

Resampling Thresholds To address the weight degeneracy problem, where the variance of the importance weights increases over time, we apply a resampling step.

- **Effective Sample Size (ESS):** We monitor $ESS = (\sum(W_n^i)^2)^{-1}$ to assess weight variability, i.e. how many samples are in effect.
- **Adaptive Resampling:** Resampling is triggered only when $ESS < N_T$, typically with $N_T = N/2$.
- **Systematic Resampling:** We utilize systematic resampling due to its ease of implementation and lower variance compared to multinomial schemes.

1.2.5 Experiments

The filters are evaluated using simulated data with SVSSM parameters $\alpha = 0.91, \sigma = 1.0, \beta = 0.5$.

Relevant Metrics

Performance is evaluated across the following dimensions:

- **Root Mean Squared Error (RMSE)**
- **Effective Sample Size (ESS):** Quantifies the health of the particle set and the frequency of resampling.
- **Thrashing:** Resampling can be costly and if it occurs too often via particle degeneracy, the system may be inefficient.
- **Computational Efficiency:** We track runtime and peak CPU/GPU memory to evaluate scalability for real-time banking applications.
- **Accuracy-Efficiency Trade-off** More particles results in better estimates. This comes at a tradeoff in resource usage.

Results

Within our experiments, we compare UKF, EKF and PF with various resampling thresholds and particle numbers. The summary table is presented here:

Table 1.2: Performance Comparison Summary of Non-Linear Filters on SVSSM

Method	RMSE	Time (s)	Mem (KB)	Avg ESS
EKF	1.1660	0.0482	112.5	—
UKF	1.2126	0.0768	32.0	—
PF ($N = 50$, $Th = 0.2$)	1.2027	0.2509	42.5	20.7
PF ($N = 50$, $Th = 0.5$)	1.1399	0.2336	45.0	30.2
PF ($N = 50$, $Th = 0.9$)	1.2365	0.2653	43.8	35.9
PF ($N = 200$, $Th = 0.2$)	1.1216	0.2461	42.1	77.6
PF ($N = 200$, $Th = 0.5$)	1.1545	0.2711	48.6	119.5
PF ($N = 200$, $Th = 0.9$)	1.1520	0.2645	41.5	143.9
PF ($N = 1000$, $Th = 0.2$)	1.1437	0.2343	41.6	378.8
PF ($N = 1000$, $Th = 0.5$)	1.1479	0.2417	41.7	593.0
PF ($N = 1000$, $Th = 0.9$)	1.1521	0.2556	43.4	719.3

Filter Accuracy and Linearization The EKF achieves a decent RMSE (1.1660) but relies on local linearization that can fail under extreme volatility spikes. Surprisingly, the UKF exhibits a higher RMSE (1.2126) in this specific trial-set, suggesting that while sigma-point methods capture higher-order moments, they still struggle with the asymmetric nature of exponential observation noise.

The Price of Flexibility As noted in the literature, the flexibility of particle methods comes at a significant computational cost [4]. The PF variants consistently require longer execution times (0.23-0.27s) compared to EKF (0.048s). This reflects the overhead of propagating N samples and the additional logic required for resampling.

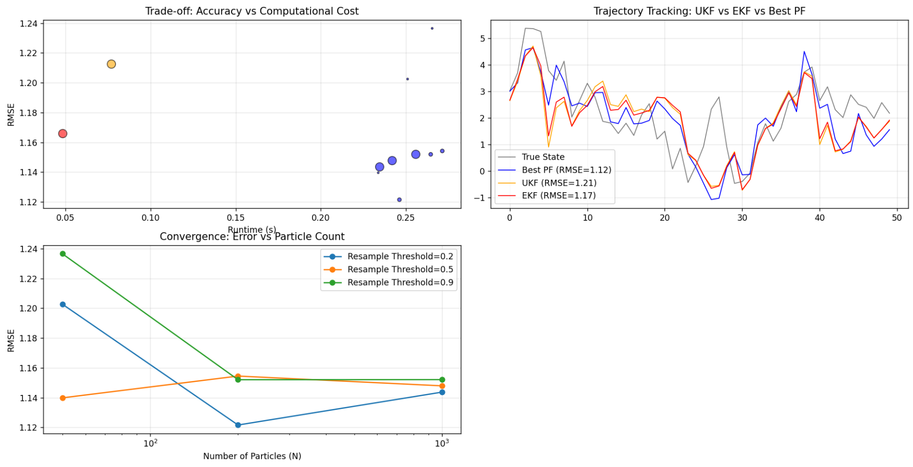

Figure 1.2: Experiment Summary

Resampling Dynamics and ESS The results in Table 1.2 and Figure 1.2 highlight the critical role of the resampling threshold (Th) and particle count (N):

- **Weight Degeneracy:** Low thresholds ($Th = 0.2$) result in lower average ESS (e.g. 20.7 for $N = 50$), indicating that the filter is carrying many particles with negligible weight.
- **Sample Impoverishment:** While higher thresholds ($Th = 0.9$) maintain a high ESS, they trigger resampling more frequently. This resets the system to prevent variance explosion but can lead to path degeneracy, where only a few unique particle lineages survive.
- **Consistency:** As N increases to 1000, the estimates become more stable and consistent with the target distribution and is consistent with [16] as the Monte Carlo variance decreases at a rate of $\mathcal{O}(1/N)$.

- **Low Threshold ($Th = 0.2$):** Minimizes computational overhead but risks weight degeneracy, where the effective sample size (ESS) collapses, leaving only a few particles to represent the distribution.
- **High Threshold ($Th = 0.9$):** Maintains a high average ESS (719.3 for $N = 1000$) by resetting the system frequently. However, excessive resampling can lead to sample impoverishment, reducing the diversity of particle lineages.

Numerical Stability and Memory The memory footprint remains relatively stable across PF configurations (41-48 KB). This stability is expected for a filtering task where only the terminal-value particles X_n^i must be stored to maintain a sequential algorithm.

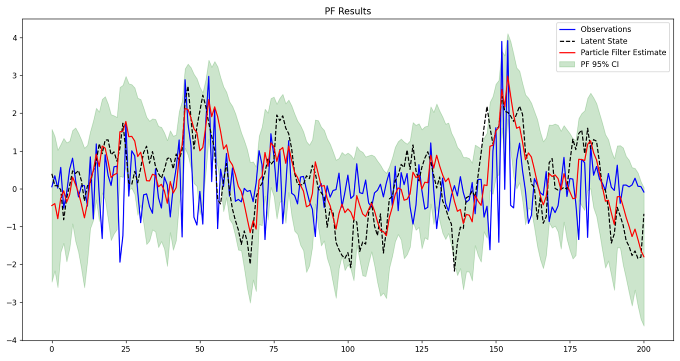

Figure 1.3: Particle Filtering Tracking Visualization (N=1000, Th=0.5)

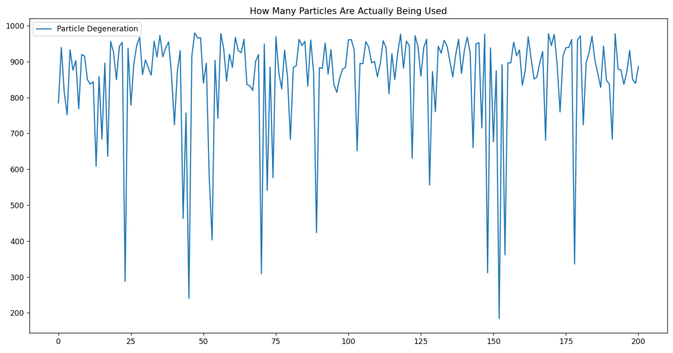

Figure 1.4: Particle Degeneracy Visuals (N=1000, Th=0.5)

Particle Degeneracy Visualization We briefly comment on the particle degeneracy encountered during the experiments in Figure 1.4. The weight concentration is noted to increase, i.e. low ESS, upon encountering volatility spikes in the observation. This can be viewed in the graphs above.

Computational Cost The trade-off plot in 1.2 confirms that the PF variants reside in a higher computational regime (> 0.23s) compared to the EKF/UKF baselines (< 0.08s).

Convergence and Monte Carlo Variance The error analysis across varying particle counts (N) confirms that filtering performance generally improves as $N \to 1000$. This aligns with the consistency properties of SMC estimators, where the variance of the approximation error decreases at a rate of $\mathcal{O}(1/N)$ [16].

1.3 Deterministic Flows and Particle Flow Filtering

1.3.1 Lorenz96 System Equations

The Lorenz-96 model is a fundamental testbed for atmospheric data assimilation and high-dimensional filtering. It represents a coarse-grained discretization of atmospheric variables along a latitude circle, exhibiting complex chaotic behavior when the forcing parameter is sufficiently high [23].

Mathematical Formulation

For a state vector $\mathbf{x} = [x_1, x_2, \ldots, x_D]^\top$ of dimension D, the dynamics of the j-th component are governed by the following system of coupled ordinary differential equations (ODEs):

$$\frac{dx_j}{dt} = (x_{j+1} - x_{j-2})x_{j-1} - x_j + F \tag{1.14}$$

where the indices are treated cyclically such that $x_{-1} = x_{D-1}$, $x_0 = x_D$ and $x_{D+1} = x_1$. The constant F represents an external forcing term; a value of $F = 8$ is typically chosen to induce a fully chaotic regime characterized by positive Lyapunov exponents.

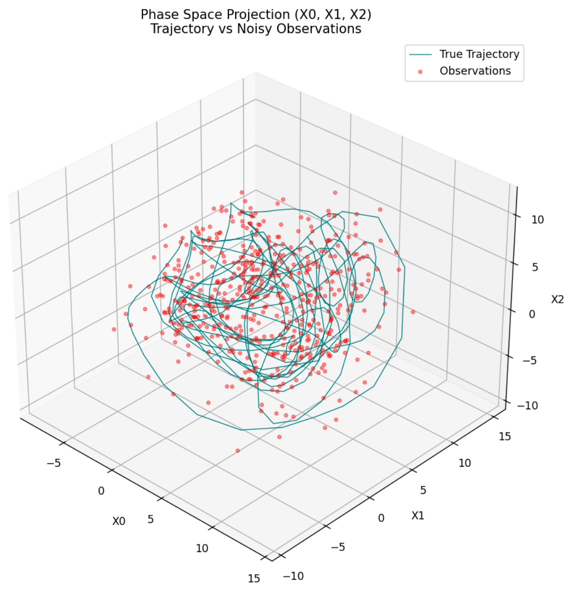

Figure 1.5: Visualizing the Lorenz96 Model Observations and True States

Relevance to Particle Flow Filtering

The Lorenz-96 model presents three primary challenges that motivate the use of deterministic and kernel flows over standard Particle Filters.

- **Curse of Dimensionality:** As D increases (e.g., $D = 40$), the volume of the state space grows exponentially. Standard PFs suffer from immediate weight collapse, as the likelihood $g(y|x)$ becomes extremely peaked in high dimensions, making it nearly impossible for random samples to fall in high-probability regions.
- **Sparse Observations:** In practice, only a subset of the D variables are observed (e.g., every second or fourth variable). The filter must therefore rely on the coupled non-linear dynamics to infer the latent states of the unobserved components.

Filtering Implementation

In our experiments, the continuous-time dynamics are integrated using a fourth-order Runge-Kutta (RK4) scheme. For the flow implementation, the Jacobian $\nabla h(x)$ is computed at each pseudo-time step λ to drive the particles toward the manifold defined by the sparse observations. We monitor the conditioning number of the state covariance P to ensure the numerical stability of the flow integration, particularly as the system dimension D scales.

1.3.2 Exact Daum-Huang Flow

The Exact Daum-Huang (EDH) flow, introduced by Daum and Huang, defines a vector field that moves particles from a prior distribution to a posterior distribution in continuous pseudo-time $\lambda \in [0, 1]$ [10, 24]. This flow is derived by solving a homotopy between the log-prior and the log-posterior densities, governed by a partial differential equation (PDE). For a Linear-Gaussian system, the EDH flow is defined by the ODE:

$$\frac{dx}{d\lambda} = f(x, \lambda) = A(\lambda)x + b(\lambda)$$

where the matrix $A(\lambda)$ and vector $b(\lambda)$ are determined by the state covariance P and observation matrix H. Specifically, for a zero-mean measurement noise with covariance R, the flow parameters are computed as:

$$A(\lambda) = -\frac{1}{2}PH^\top(\lambda HPH^\top + R)^{-1}H \tag{1.15}$$

$$b(\lambda) = (I + 2\lambda A(\lambda))\left[(I + \lambda A(\lambda))PH^\top R^{-1}y + A(\lambda)\bar{x}\right] \tag{1.16}$$

where $\bar{x}$ is the prior mean and y is the current observation. By integrating this flow, particles are deterministically shifted to the mode of the measurement update without the need for discrete resampling.

Local Exact Daum-Huang

The Local Exact Daum-Huang (LEDH) flow extends the EDH framework to non-linear systems by linearizing the observation model $h(x)$ locally around each particle [24]. Unlike the global EDH, which assumes a single Jacobian for the entire particle cloud, LEDH computes a unique vector field for each particle i based on the Jacobian $H_i = \nabla h(x^i)$. The flow equation for the i-th particle remains:

$$\frac{dx^i}{d\lambda} = A^i(\lambda)x^i + b^i(\lambda) \tag{1.17}$$

where $A^i(\lambda)$ is now particle-dependent, calculated using the local Jacobian H_i in place of a global H. This local adaptation allows the flow to better capture the curvature of complex manifolds, though it requires significantly more computational resources to calculate separate flow matrices and integrate N distinct ODEs.

1.3.3 Invertible Particle Flow Particle Filtering

The Invertible Particle Flow Particle Filter (PF-PF) addresses a critical theoretical limitation of standard deterministic flows: the loss of exactness in the importance weights [9]. Traditional particle flow filters often discard importance weights or assume they are uniform after the flow, which can introduce significant bias in non-linear dynamics. The PF-PF framework formalizes the flow as an invertible mapping $\mathcal{T} : \mathbb{R}^d \to \mathbb{R}^d$ that acts as a diffeomorphism between the prior and posterior distributions.

Theoretical Framework and Weight Update

By defining the flow as a continuous-time transformation, the PF-PF leverages the change-of-variables formula to compute the effective proposal density induced by the flow [9]. To ensure the filter remains asymptotically consistent, the importance weights must be updated to account for the local volume deformation of the particle space. The weight update equation is given by:

$$w^i_{post} \propto w^i_{prior} \frac{p(y|x^i_{post})p(x^i_{post}|x_{t-1})}{q(x^i_{post}|x_{t-1}, y)} \left|\det \nabla \mathcal{T}(x^i_{prior})\right|^{-1} \tag{1.18}$$

where $|\det \nabla \mathcal{T}|$ is the absolute value of the determinant of the Jacobian of the flow mapping. This term compensates for the spatial concentration or expansion of particles during the migration step.

Numerical Implementation: Jacobian Tracking

Implementing the PF-PF requires the simultaneous integration of the state ODE and the evolution of the Jacobian matrix Ψ along the pseudo-time $\lambda \in [0, 1]$ [9]. To maintain numerical stability and avoid overflow, the log-determinant is typically tracked using the trace identity:

$$\frac{d}{d\lambda} \ln |\det \Psi(\lambda)| = \mathrm{Tr}\left(\Psi(\lambda)^{-1} \frac{d\Psi(\lambda)}{d\lambda}\right) \tag{1.19}$$

where $\frac{d\Psi}{d\lambda}$ is the derivative of the vector field with respect to the state.

Advantages in High-Dimensional Inference

By maintaining invertibility, in theory, PF-PF formulation allows the filter to eliminate weight collapse. By moving particles toward the likelihood mode, the filter maintains a high ESS even in high-dimensional spaces.

PFPF-EDH and PFPF-LEDH Implementations

The PF-PF framework can be specialized into two distinct algorithmic variants based on the underlying flow computation

PFPF-EDH (Global Flow) The **PFPF-EDH** variant utilizes the global Exact Daum-Huang flow formulation, typically derived for systems where the observation model can be globally approximated as linear-Gaussian.

- **Vector Field:** The particles are migrated according to a single, shared vector field $\frac{dx}{d\lambda} = A(\lambda)x + b(\lambda)$.
- **Computational Efficiency:** Because the matrix $A(\lambda)$ and vector $b(\lambda)$ are computed globally for the entire particle cloud, this method is significantly faster than local variants.

PFPF-LEDH (Local Flow) The **PFPF-LEDH** variant is designed for systems with high non-linearity or non-Gaussianity where a global linearization is insufficient. This is analogous to the difference between EDH and LEDH.

- **Local Adaptation:** Each particle i follows a unique trajectory driven by a local Jacobian $H_i = \nabla h(x^i)$, allowing the flow to adapt to the local curvature of the observation manifold.
- **Weight Consistency:** The local nature of the flow requires N individual Jacobian determinant calculations, $|\det \nabla \mathcal{T}(x^i)|$, ensuring that importance weights remain consistent across complex state-space topographies.
- **Robustness:** While computationally demanding, the local flow provides a more robust proposal distribution, effectively preventing weight collapse in scenarios where the posterior distribution is heavily distorted.

1.3.4 Kernel-Embedded Particle Flow Filter

For extremely high-dimensional systems where computing Jacobians is numerically unstable or computationally prohibitive, the Kernel-Embedded Particle Flow Filter (kernel PFF) offers a non-parametric alternative [3]. Instead of relying on a Taylor expansion of the observation model, KEPFF embeds the particle distribution into a Reproducing Kernel Hilbert Space (RKHS) [25].

RKHS Formulation

The core innovation of the kernel PFF is to approximate the optimal flow drift, $f(x, \lambda)$, as a function within an RKHS, minimizing the Kullback-Leibler (KL) divergence between the transported particle distribution and the target posterior [3]. The solution for the drift takes the form of a weighted sum of kernel functions centered at the particles:

$$f(x) = \sum_{j=1}^{N} K(x, x^j) c_j \tag{1.20}$$

where $K(\cdot, \cdot)$ is the kernel and c_j are coefficient vectors determined by solving a linear system. This formulation skips the explicit calculation of the Jacobian $\nabla h(x)$, reducing the complexity significantly for high-dimensioned SSMs.

Matrix-Valued Kernels and Marginal Collapse

A finding in [**?**] is that standard scalar kernels are insufficient for high-dimensional systems. Scalar kernels enforce a uniform correlation structure that often leads to marginal collapse, where the variance of the observed variables is underestimated while unobserved variables remain uncorrected. To address this, the kernel PFF employs Matrix-Valued Kernels $K(x, x') = D \cdot k(x, x')$, where D is a matrix that allows for component-specific scaling.

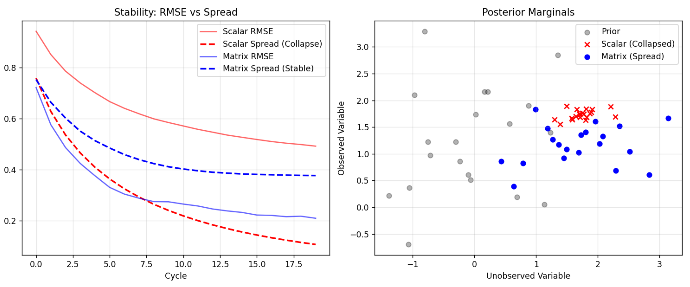

Figure 1.6: Observe posterior collapse within the scalar kernel vs. the stability of the matrix kernel PF within a 1000-dimensional Lorenz96 SSM

1.3.5 Experiments

The experiments conducted within this section are to assess filter performance in the Lorenz96 SSM.

Relevant Metrics

To evaluate the accuracy, stability and efficiency of the flow-based filters, we track the following metrics:

- **RMSE (Root Mean Square Error)**
- **OMAT (Optimal Mass Transport):** Represents the magnitude of the flow vector field integrated over pseudo-time. A higher OMAT value indicates a more aggressive migration of particles, which often correlates with numerical stiffness in the ODE solver.
- **Covariance Conditioning:** The condition number of the state covariance matrix P. High condition numbers ($\cdot \gg 10^3$) indicate numerical instability and potential singularity in the matrix inversion steps required for EDH/LEDH.
- **ESS (Effective Sample Size)**
- **Runtime and Memory**

EDH vs. LEDH vs. PFPF

We first evaluate the impact of the invertible flow correction across varying dimensions to assess how well the filters mitigate bias compared to unweighted deterministic flows.

Low-Dimensional Performance (Dim=10) As shown in Table 1.3, the Invertible PFPF variants demonstrate superior accuracy in lower dimensions. The PFPF_EDH achieves an RMSE of 0.4321, a significant reduction compared to the standard EDH (0.5834). This confirms that the standard EDH flow introduces a systematic bias by discarding importance weights, which the PFPF corrects via the Jacobian determinant update. Furthermore, the OMAT values for PFPF (≈ 1.5) are less than half that of standard EDH (3.28), suggesting that weighted particles require less physical migration to represent the posterior, resulting in a smoother flow.

Scalability and Degeneracy (Dim=50) Table 1.4 illustrates the performance as dimensionality increases. While PFPF variants maintain superior accuracy (RMSE ≈ 0.48) compared to deterministic baselines (RMSE > 0.59), they suffer from weight degeneracy. The **Min ESS** for PFPF_LEDH drops to 2.8, highlighting a critical trade-off: eliminating bias comes at the cost of sample impoverishment in high dimensions, necessitating frequent resampling.

Table 1.3: Performance Comparison of EDH, LEDH and PFPF, Dim=10, particle num=100

Method	**RMSE**	**OMAT**	**Avg ESS**	**Min ESS**	**Time (s)**	**Mem (MB)**
EDH	0.5834	3.2786	100.0	100.0	7.3001	3.81
LEDH	0.5352	1.7814	100.0	100.0	7.0857	3.80
PFPF_EDH	0.4321	1.5015	52.8	9.1	8.6790	4.68
PFPF_LEDH	0.4627	1.5013	59.7	17.2	8.3868	4.56

Table 1.4: Performance Comparison of EDH, LEDH and PFPF, dim=50, particle num=100

Method	**RMSE**	**OMAT**	**Avg ESS**	**Min ESS**	**Time (s)**	**Mem (MB)**
EDH	0.6264	8.1939	-	-	17.4233	3.81
LEDH	0.5988	4.4015	-	-	17.6325	3.80
PFPF_EDH	0.4789	3.5839	31.9	6.3	19.6224	4.67
PFPF_LEDH	0.4932	3.6183	35.8	2.8	19.2404	4.55

Kernel PFF vs. EDH vs. LEDH

Table 1.5 demonstrates the resilience of the kernel PFF. As the dimension scales to 100, the deterministic flows degrade significantly, with LEDH yielding the worst RMSE (1.5900), likely due to the failure of local linearization in sparse observation regimes. In contrast, the kernel flow maintains the best accuracy (1.3622) and the lowest condition number ($\kappa = 1.46$). This validates that RKHS embeddings effectively circumvent the numerical ill-conditioning ($\kappa > 2.5$) associated with explicit Jacobian inversion.

Table 1.5: Metrics Summary: Comparison of Deterministic and Kernel Flows

Method	**Dim50 RMSE**	**Conditioning**	**OMAT**	**Dim100 RMSE**
EDH	0.9874	2.55	5.63	1.5066
LEDH	0.9068	2.87	5.29	1.5900
Kernel	0.7017	1.46	2.21	1.3622

Chapter 2

Part 2 - Stochastic Particle Flow and Differentiable PF

2.1 Stochastic Particle Flow

Stochastic Particle Flow filters generalize the deterministic flow framework by modelling the transport of probability mass via an SDE rather than a pure ODE, unlike the EDH/LEDH [5, 15].

2.1.1 Mathematical Formulation of Stochastic PF

Let $\pi(\lambda)$ denote the intermediate probability density connecting the prior π_0 and the posterior π_1. The evolution of the particle state x_λ is governed by the Ito SDE:

$$dx_\lambda = \mu(x_\lambda, \lambda)d\lambda + \sigma(\lambda)dW_\lambda \tag{2.1}$$

where W_λ is a standard Wiener process.

The drift term $\mu(x, \lambda)$ is derived to satisfy the Fokker-Planck equation associated with the desired homotopy $\pi(\lambda)$. For a Linear Gaussian system, the optimal drift takes the form:

$$\mu(x, \lambda) = A(\lambda)x + b(\lambda) \tag{2.2}$$

Crucially, the diffusion term $\sigma(\lambda) \neq 0$ alters the solution for $A(\lambda)$. Unlike the deterministic case ($\sigma = 0$), the stochastic drift does not need to account for the entire probability mass transport via advection alone; the diffusion term contributes to the spreading of the density.

2.1.2 Stiffness Mitigation with Stochastic PF

Numerical stiffness in particle flow arises when the drift magnitude $||\mu(x, \lambda)||$ becomes large, causing the Jacobian of the flow to become ill-conditioned. The introduction of the diffusion term $\sigma(\lambda)$ mitigates this via two mechanisms [5]:

- **Drift Relaxation:** By satisfying the Fokker-Planck equation with both drift and diffusion components, the magnitude of the required drift vector is reduced. Mathematically, the flow velocity is inversely related to the diffusion strength; as $\sigma(\lambda)$ increases, the L_2 norm of the drift $||\mu||_2$ decreases.

- **Conditioning of the Flow Matrix:** In the EDH/LEDH framework, the matrix $A(\lambda)$ involves the inversion of terms like $(\lambda HPH^\top + R)$. The stochastic formulation effectively adds a regularization term to this inversion, preventing the condition number $\kappa(A(\lambda))$ from exploding when the measurement noise covariance R is small.

2.1.3 LEDH-PFPF with Stochastic Flow

Integrating the optimized stochastic flow of [5] into the invertible PF-PF of [9] provides a way to stabilize the importance weights.

Jacobian Regularization in Weight Update

The PF-PF weight update relies on the determinant of the Jacobian of the mapping $\mathcal{T}$:

$$w_{post} \propto w_{prior} \cdot \frac{p(y|x)p(x)}{q(x)} \cdot |\det \nabla \mathcal{T}|^{-1} \tag{2.3}$$

In a fully deterministic LEDH flow, the Jacobian $\nabla \mathcal{T}$ can become singular or highly skewed (large condition number) if the flow is stiff, leading to unstable determinants (≈ 0 or $\gg 1$).

By using the stochastic flow as the proposal mapping:

1. The map $\mathcal{T}$ becomes the solution to the SDE, which is smoother due to the diffusion regularization.
2. The Jacobian $\Psi = \nabla \mathcal{T}$ remains better conditioned ($\kappa(\Psi) \approx 1$), ensuring that the determinant term $|\det \Psi|^{-1}$ does not fluctuate wildly.
3. This results in a lower variance of the importance weights w_{post} and a consistently higher ESS.

2.1.4 Experiments

Relevant Metrics

- **RMSE (Root Mean Square Error)**
- **OMAT (Optimal Mass Transport)**
- **Covariance Conditioning**
- **ESS (Effective Sample Size)**
- **Runtime and Memory**

Replicating the Results in Dai (2021)

We first validate the claims regarding stiffness mitigation by comparing the standard linear homotopy against the optimized stochastic homotopy proposed by Dai (2021). Table 2.1 presents the Mean Squared Error (MSE) and the Trace of the Covariance Matrix ($\mathrm{Tr}(P)$) over 20 independent runs.

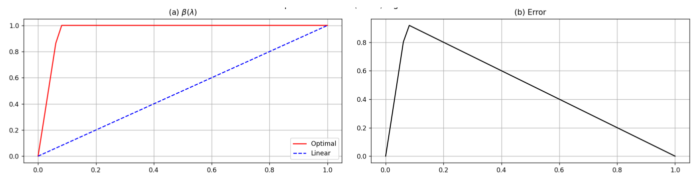

Figure 2.1: Replicating the results of Dai (2021)

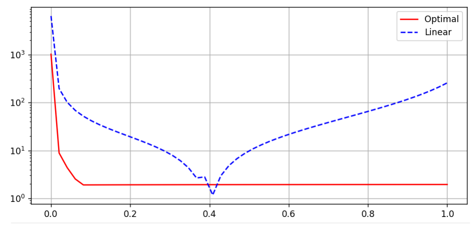

Figure 2.2: Comparing stiffness of the optimal homotopy with the linear homotopy

Analysis of Instability: The results reveal the severe instability inherent in the linear homotopy. While some runs (e.g., Run 4, 16) achieve low error, others suffer from catastrophic divergence. Notably, Run 17 exhibits an explosion in error ($MSE \approx 3.4 \times 10^5$) and covariance trace ($\approx 2.7 \times 10^6$), skewing the average significantly. This confirms that without stiffness mitigation, the flow can force the covariance matrix into ill-conditioned states, leading to filter divergence.

Effectiveness of Optimization: In contrast, the Optimized Stochastic Flow yields consistent performance across all 20 runs. The average $\mathrm{Tr}(P)$ is reduced by two orders of magnitude ($148,525$ vs $8,240$), indicating that the optimized diffusion term successfully regularizes the flow, keeping the particle cloud tightly bound around the true posterior and preventing the numerical stiffness that causes covariance explosion.

Table 2.1: Performance Comparison: Linear vs. Optimized Stochastic Flow (20 Runs)

Run	$\mathbf{MSE_{lin}}$	$\mathbf{MSE_{opt}}$	$\mathbf{Tr(P)_{lin}}$	$\mathbf{Tr(P)_{opt}}$
1	5205.2005	198.3045	7545.93	43.28
2	85.9761	198.2279	6654.23	32.49
3	40.9802	239.9906	39794.80	30.99
4	6.6711	82.8081	4323.86	9003.98
5	33.3798	215.8495	210.34	29.51
6	27.4787	248.3737	365.26	42.14
7	26.6946	222.4691	277.62	42.00
8	13.4264	258.8133	749.09	26.69
9	38.6290	199.6714	1062.30	11.23
10	2.4624	189.5855	186243.77	19.61
11	26.1993	261.4333	577.38	38.97
12	54.2508	205.2732	1141.70	16.40
13	24.4706	204.8470	437.02	27.99
14	206.1767	192.2372	980.33	48.05
15	6.2588	189.9924	10994.00	29.20
16	1.1409	29872.1778	954.55	102139.82
17	342339.2538	3302.1500	2703673.23	53045.75
18	6.3398	188.6266	894.42	57.00
19	26.0099	214.9586	832.35	91.45
20	15.7185	188.2900	2806.83	25.39
Avg	**17409.3359**	**1843.7040**	**148525.95**	**8240.10**

Implementing Stochastic Flow in LEDH-PFPF

Next, we evaluate the hybrid filter where the optimized stochastic flow [5] is used as the proposal mechanism within the Invertible PF-PF framework [26]. **We note that within our experiments, the time to execute for the stochastic flow PF-PF is much higher than the baseline filter as we computed the inverse matrix using the CPU as opposed to GPU due to crashes in the TF GpuSolver from the operation.** We compare both PF-PF algorithms within the Lorenz96 environment used in the previous sections.

Low-Dimensional Performance (Dim=10) Table 2.2 demonstrates that replacing the deterministic LEDH proposal with the stochastic flow improves accuracy by approximately 15% (RMSE goes from 0.6252 to 0.5340).

- **Improved Conditioning:** The average condition number drops from 34.23 to 21.29. This indicates that the stochastic flow generates a smoother mapping with a more stable Jacobian, which directly benefits the PF-PF weight update step.
- **Stiffness Reduction:** The lower OMAT score (1.69 vs 1.95) confirms that the stochastic particles require less aggressive transport to reach the high-probability region.

Table 2.2: Performance Comparison: LEDH-PFPF vs. Stoch-PFPF (Dim=10)

Method	**RMSE**	**OMAT**	**Avg Cond**	**Avg ESS**	**Time (s)**	**Mem (MB)**
LEDH_PFPF	0.6252	1.9536	34.23	56.5	4.6698	5.28
Stoch_PFPF	0.5340	1.6986	21.29	58.0	50.2979*	5.30

Table 2.3: Performance Comparison: LEDH-PFPF vs. Stoch-PFPF (Dim=100)

Method	**RMSE**	**OMAT**	**Avg Cond**	**Avg ESS**	**Time (s)**	**Mem (MB)**
LEDH_PFPF	0.6422	6.5178	48.00	25.4	4.7596	5.27
Stoch_PFPF	0.5960	6.0095	29.69	26.8	160.1749*	5.29

High-Dimensional Performance (Dim=100) In the 100-dimensional setting (Table 2.3), the accuracy gap persists, with the Stoch-PFPF achieving an RMSE of 0.5960 compared to 0.6422 for the standard LEDH-PFPF.

- **Conditioning vs. ESS:** While both methods suffer from sample impoverishment (ESS dropping to ≈ 25), the stochastic flow maintains significantly better matrix conditioning (Avg Cond = 29.69 vs 48.00). This suggests that even in sparse high-dimensional regimes, the stochastic regularization prevents the flow matrices from approaching singularity.

2.2 Differentiable Particle Filtering

2.2.1 The DPF Objective

Consider non-linear SSM defined over a latent state sequence $x_{0:T} := \{x_0, \ldots, x_T\}$ and an observation sequence $y_{1:T} := \{y_1, \ldots, y_T\}$. The model is governed by the following transition and observation densities, parameterized by θ [27]:

$$x_t \sim p_\theta(x_t | x_{t-1}) \tag{2.4}$$

$$y_t \sim p_\theta(y_t | x_t) \tag{2.5}$$

The goal of the particle filter is to approximate the posterior distribution $p(x_{0:t}|y_{1:t})$ sequentially. At time $t-1$, this posterior is represented by a set of N weighted particles $\{(x^i_{t-1}, w^i_{t-1})\}_{i=1}^N$, such that the empirical measure approximates the true density:

$$p(x_{0:t-1}|y_{1:t-1}) \approx \sum_{i=1}^{N} w^i_{t-1} \delta_{x^i_{0:t-1}}(x_{0:t-1}) \tag{2.6}$$

where δ_x is the Dirac delta function centered at x and the weights are normalized such that $\sum_{i=1}^N w^i_{t-1} = 1$.

In a standard DPF setting, we aim to optimize the parameter set θ (which may include neural network weights governing the proposal $q_\phi(x_t|x_{t-1}, y_t)$ and measurement models) by minimizing a loss function $\mathcal{L}(\theta)$ via gradient descent [11]. The primary obstacle is the non-differentiable nature of the discrete resampling step, which prevents the backpropagation of gradients from time t to $t-1$.

2.2.2 Soft-Resampling

The core limitation of traditional resampling is its discrete selection mechanism. If we define the resampling operation as selecting indices $a_t^i \sim \text{Cat}(w_{t-1}^{1:N})$, the resulting particle $x_t^i = x_{t-1}^{a_t^i}$ has zero gradient with respect to the source particle states x_{t-1}^j because the index selection is a step function.

Convex Combination Formulation

Soft-resampling addresses this by relaxing the hard selection into a continuous weighted average. Instead of selecting a single ancestor, each new particle $\tilde{x}_t^i$ is formed as a convex combination of the entire set of particles from the previous time step:

$$\tilde{x}_t^i = \sum_{j=1}^{N} \pi_{i,j} x_{t-1}^j, \quad \text{for } i = 1, \ldots, N \tag{2.7}$$

Here, $\Pi = (\pi_{i,j}) \in \mathbb{R}^{N \times N}$ is a transport matrix where $\pi_{i,j}$ represents the probability (or mass) transferred from source particle j to target particle i. For the operation to be valid, Π must be row-stochastic:

$$\sum_{j=1}^{N} \pi_{i,j} = 1, \quad \forall i \in \{1, \ldots, N\} \tag{2.8}$$

Under this formulation, the gradient $\nabla_{x_{t-1}} \mathcal{L}$ can flow backward through the linear combination, enabling end-to-end training of the networks generating the states [27].

The Particle Transformer Mechanism

The particle transformer mechanism is necessary to implement soft-resampling [27]. It leverages the attention mechanism prevalent in deep learning to parameterize the transport matrix Π. Let each particle x_{t-1}^j be projected into a key vector K_j and a value vector V_j (i.e. the state itself), while the target slots are represented by learned query vectors Q_i. The transport weights are computed as:

$$\alpha_{i,j} = \frac{\exp\left(\frac{Q_i^\top K_j}{\sqrt{d_k}} + \log w_{t-1}^j\right)}{\sum_{k=1}^{N} \exp\left(\frac{Q_i^\top K_k}{\sqrt{d_k}} + \log w_{t-1}^k\right)} \tag{2.9}$$

where d_k is the dimension of the key vectors.

Note that the term $\log w_{t-1}^j$ acts as a bias in the attention score. As $w_{t-1}^j \to 0$, the term $\log w_{t-1}^j \to -\infty$, effectively suppressing the contribution of particles with low importance weights. Conversely, particles with high likelihoods dominate the softmax, ensuring that the new set of particles $\{\tilde{x}_t^i\}$ is concentrated around the high-probability regions of the posterior.

Statistical Implications and Limitations

While soft-resampling restores differentiability, it alters the statistical variance of the particle set.

- **Variance Reduction:** Since every new particle is a weighted average of old particles, the variance of the new set is strictly less than or equal to the variance of the source set (by Jensen's inequality) [11, 27]. Repeated application can lead to a posterior collapse where all particles converge to a single point, usually the meean.

- **Mode Merging:** In multi-modal posteriors, averaging particles from distinct modes can result in new particles located in low-probability regions between modes, violating the physical constraints of the system.

2.2.3 Entropy-Regularized Optimal Transport (Sinkhorn)

To mitigate the mode-merging behavior of soft-resampling while maintaining differentiability, [11] propose formulating resampling as an Optimal Transport (OT) problem. The goal is to find a transport plan that transforms the current weighted measure $\mu = \sum w_{t-1}^i \delta_{x_{t-1}^i}$ into the target uniform measure $\nu = \sum \frac{1}{N} \delta_{x_{t-1}^i}$ with minimal cost [28].

Optimization Formulation

Let $C \in \mathbb{R}^{N \times N}$ be a cost matrix where $C_{ij} = c(x_{t-1}^i, x_{t-1}^j)$ represents the effort to move mass from particle i to particle j (e.g. Euclidean distance). The standard Kantorovich OT problem is [29, 30]:

$$\min_{P \in U(w,\mathbf{u})} \sum_{i,j} P_{ij} C_{ij} \tag{2.10}$$

where the feasible set $U(w, u)$ is the set of non-negative matrices with row sums equal to the source weights $w = [w_{t-1}^1, \ldots, w_{t-1}^N]^\top$ and column sums equal to the uniform target $u = [\frac{1}{N}, \ldots, \frac{1}{N}]^\top$.

The solution to the exact OT problem is often a sparse permutation matrix via hard resampling, which remains non-differentiable. To smooth the solution, an entropy regularization term $H(P) = -\sum_{i,j} P_{ij}(\log P_{ij} - 1)$ is introduced [11]:

$$P_\epsilon^* = \arg \min_{P \in U(w,u)} \langle P, C \rangle - \epsilon H(P) \tag{2.11}$$

where $\epsilon > 0$ is a regularization coefficient.

The Sinkhorn-Knopp Algorithm

The solution to the regularized problem has a unique form given by the scaling of the Gibbs kernel $K_{ij} = \exp(-C_{ij}/\epsilon)$ [31]:

$$P_{ij}^* = u_i K_{ij} v_j \tag{2.12}$$

The vectors u and v are scaling factors that ensure the marginal constraints are met. They can be found using the fixed-point iterations known as the Sinkhorn algorithm:

$$v^{(k+1)} = \frac{u}{K^\top u^{(k)}} \tag{2.13}$$

$$u^{(k+1)} = \frac{w}{K v^{(k+1)}} \tag{2.14}$$

where the division is element-wise.

Differentiability and Gradients

In the context of DPFs, this algorithm allows for the computation of gradients via unrolling. The Sinkhorn iterations (between $L = 10$ and 50 steps) define a computational graph consisting of differentiable matrix multiplications and element-wise divisions.

- **Forward Pass:** We compute the optimal transport plan P_ϵ^*. The resampled particles are then defined as $\tilde{x}_t = N \cdot P_\epsilon^{*\top} x_{t-1}$. Note that unlike soft-resampling, this transport tends to preserve the geometric structure of the cloud.
- **Backward Pass:** Gradients $\nabla_\theta \mathcal{L}$ can flow back through P_ϵ^* to the cost matrix C and consequently to the particle positions x_{t-1}.
- **Role of ϵ:** The parameter ϵ acts as a temperature. As $\epsilon \to 0$, P_ϵ^* converges to the sharp, deterministic resampling assignment (low bias, high variance gradients). As $\epsilon \to \infty$, P_ϵ^* approaches the independent product of marginals (high bias, low variance).

2.2.4 Other DPF Algorithms

We briefly mention a few other DPF algorithms for comparison in the next section.

Optimal Placement Resampling

While Entropy-Regularized Optimal Transport smooths the resampling step by optimizing a transport plan, Optimal Placement Resampling takes a different approach: it optimizes the locations of the new particles directly, rather than their weights or assignment probabilities [32].

In this framework, the goal is to find a set of N new particle locations $\{z^j\}_{j=1}^N$ that minimize the Wasserstein distance to the current empirical measure $\mu = \sum_{i=1}^N w_t^i \delta_{x_t^i}$. This is equivalent to finding a transport plan P and locations Z that minimize:

$$\min_{Z, P \in U(w, \frac{1}{N})} \sum_{i,j} P_{ij} \|x_t^i - z^j\|^2 \tag{2.15}$$

Unlike standard resampling where z^j must be a copy of some x_t^i, here z^j can be any point in the state space.

- **Mechanism:** This method effectively performs a form of k-means clustering on the weighted particles, where the new particles z^j move to the centroids of the probability mass.
- **Differentiability:** Since the optimal locations z^j are continuous functions of the source particles x_t^i and weights w_t^i, gradients can flow directly through the coordinates of the new particles.
- **Advantage:** This approach allows for geometric adaptation, placing particles in high-density regions even if no single previous particle was exactly at that location, potentially reducing the variance of the state estimator more effectively.

Conditional Normalizing Flow

Standard DPFs often rely on the Bootstrap formalism [1], where the proposal distribution $q(x_t|x_{t-1}, y_t)$ is simply the dynamic model $p(x_t|x_{t-1})$. This ignores the latest observation y_t when proposing new particles, leading to inefficiency in high-dimensional or informative-observation regimes.

[33] propose integrating Conditional Normalizing Flows (CNFs) to construct observation-dependent proposal distributions.

- **Architecture:** A Normalizing Flow transforms a simple base distribution into a complex target density via a sequence of invertible, differentiable mappings f_k. In the conditional variant, these mappings are parameterized by a context vector h_t derived from the previous state x_{t-1} and the current observation y_t.
- **Proposal Construction:** The particles are generated by sampling $\epsilon \sim \mathcal{N}(0, I)$ and computing $x_t = f^{-1}(\epsilon; x_{t-1}, y_t)$. This allows the proposal to capture complex, multi-modal posteriors that simple Gaussian approximations (like EKF or UKF) cannot.
- **Training:** The flow parameters are trained end-to-end within the DPF framework. By minimizing the DPF loss, the flow learns to migrate particles into regions of high likelihood implied by the current observation, significantly improving the ESS compared to standard bootstrap proposals.

2.2.5 Experiments

We switch back to the SVSSM in order to replicate the experiments within [11,32,33] and compare each of the DPFs mentioned previously on the following metrics.

Relevant Metrics

- **RMSE (Root Mean Square Error)**
- **Differentiability (GradNorm):** The L_2 norm of the gradient of the loss with respect to a learnable initial parameter. High gradient norms often indicate high variance or numerical instability.
- **Memory**
- **Runtime**

Implementation Considerations

Implementing DPFs presents new challenges compared to prior methods. The recursive nature of the likelihood updates and the reparameterization inherent in resampling steps can lead to gradient instability. We present some techniques in our implementations that can help actualize the algorithms for practical use.

Gradient Clipping DPF techniques trained in autograd settings can exhibit exploding gradients within high-dimensional, or non-linear, settings. To prevent exploding gradients from destabilizing the filter, global norm gradient clipping is a useful technique.

Log-Space Numerics Particle filters involve the repeated multiplication of probabilities, which quickly leads to arithmetic underflow. All weight computations must be performed in the log-domain. When normalizing weights w_t^i, the operation $\frac{\exp(\log w^i)}{\sum \exp(\log w^j)}$ is numerically unstable. It must be implemented as:

$$\log \tilde{w}^i = \log w^i - \text{LogSumExp}(\log w^{1:N})$$

Cost Matrix Normalization In the Sinkhorn algorithm, the cost matrix $C_{ij} = \|x^i - x^j\|^2$ depends on the scale of the state space. If the states magnitude grows (i.e. during early training instability), C_{ij} can become arbitrarily large, causing the kernel to vanish. To prevent this, we can normalize the cost matrix at every step:

$$C_{ij} \leftarrow \frac{C_{ij}}{\max(C) + \delta}$$

This ensures the exponent $-C/\epsilon$ is stable regardless of the scale of the latent states.

Finetuning Sinkhorn-Knopp Algorithm

The performance of the Sinkhorn resampling step is governed by the entropy regularization parameter ϵ and the number of fixed-point iterations L. These parameters control a trade-off between bias (i.e. deviation from optimal transport), variance (i.e. gradient stability) and runtime.

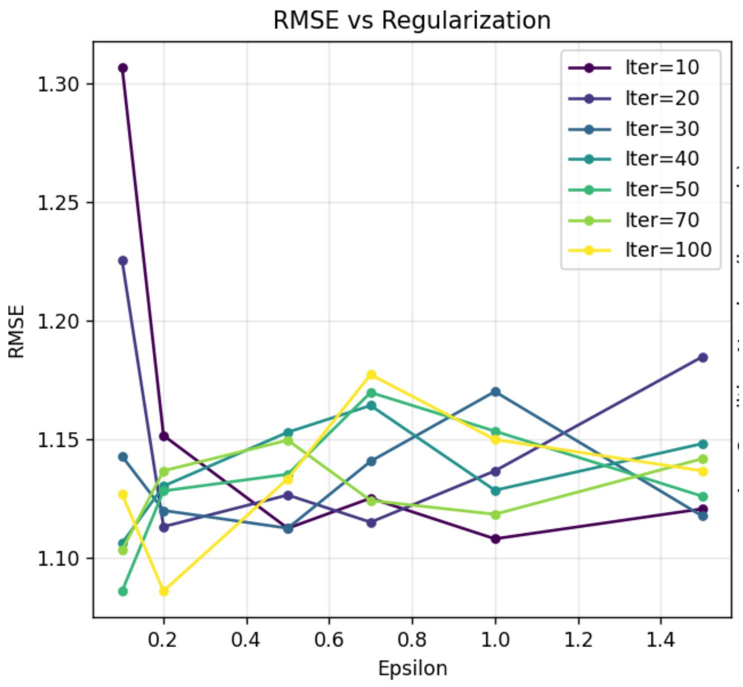

Figure 2.3: Tuning regularization within the Sinkhorn algorithm

Table 2.4 highlights two distinct regimes.

1. **Low Iteration Regime ($L = 10$):** With few iterations, a low ϵ (0.1) leads to poor convergence, resulting in high RMSE (1.3068) and exploding gradients (GradNorm ≈ 43.3). Increasing ϵ to 1.0 smooths the problem, allowing the algorithm to converge quickly to a diffuse solution, yielding better accuracy (1.1082) and stability.

2. **High Iteration Regime ($L = 50$):** Given sufficient compute time, a low ϵ (0.1) achieves the best overall accuracy (RMSE 1.0862) because it approximates the hard optimal transport plan most closely. Conversely, high ϵ (1.0) degrades performance (RMSE 1.1535) because the excessive smoothing introduces a bias, likely diffusing particles too far from the posterior peaks.

Table 2.4: Ablation Study: Effect of Entropy Regularization (ϵ) on Sinkhorn DPF Performance

Epsilon (ϵ)	**Iters**	**RMSE**	**GradNorm**	**Time (s)**
0.1	10	1.3068	43.2939	2.1030
0.2	10	1.1517	32.4348	1.4527
0.5	10	1.1126	18.1200	1.4291
0.7	10	1.1253	29.6769	1.4629
1.0	10	1.1082	22.2748	1.3854
0.1	50	1.0862	17.1973	11.9980
0.2	50	1.1283	16.8821	11.3047
0.5	50	1.1354	13.3598	11.0572
0.7	50	1.1698	5.7468	11.3635
1.0	50	1.1535	15.1861	11.1367

Runtime vs. Iterations Table 2.5 further analyzes the convergence behavior at a fixed $\epsilon = 0.2$. We observe that RMSE improves significantly when increasing iterations from 10 to 20, but plateaus or slightly degrades beyond 30 iterations. Simultaneously, the runtime increases linearly. This suggests that for practical applications, a sweet spot of 20-30 iterations provides the necessary gradient stability (GradNorm drops to ≈ 10) without incurring the cost of fully converged transport.

Table 2.5: Ablation Study: Effect of Sinkhorn Iterations on DPF Performance ($\epsilon = 0.2$)

Epsilon (ϵ)	**Iters**	**RMSE**	**GradNorm**	**Time (s)**
0.2	10	1.1517	32.4348	1.4527
0.2	20	1.1133	36.1800	2.5209
0.2	30	1.1201	10.0140	3.6295
0.2	40	1.1304	11.6826	4.8625
0.2	50	1.1283	16.8821	11.3047

Comparing Various DPFs

Table 2.6 presents a comprehensive comparison of four differentiable particle filter architectures: the entropy-regularized OT filter (Sinkhorn), Soft Resampling (Soft), Optimal Placement Resampling (OPR) and the Conditional Normalizing Flow Particle Filter (CNF-PF).

Table 2.6: Performance Comparison of DPFs on SVSSM

Method	RMSE	GradNorm	Time (s)	Mem (MB)
Sinkhorn	1.0379	1.0088	3.0586	14.7062
Soft	1.0864	2.2803	0.3588	1.0668
OPR	1.0694	1.0179	0.5815	1.5061
CNF-PF	1.0787	2.5105	0.5460	1.6401

Accuracy vs. Stability The Sinkhorn method achieves the lowest RMSE (1.0379) and the most stable gradients (GradNorm ≈ 1.0), confirming its theoretical superiority in preserving particle geometry. OPR follows closely with an RMSE of 1.0694 and similarly stable gradients (1.0179), suggesting that optimizing particle locations is a viable alternative to optimizing transport plans.

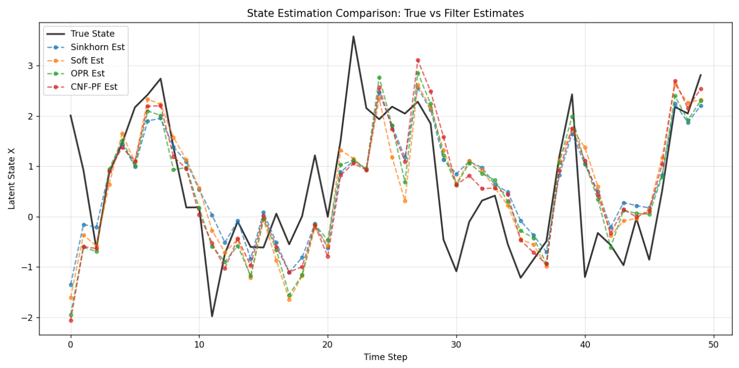

Figure 2.4: Tracking the SVSSM with the DPF algorithms

Computational Cost The stability of Sinkhorn comes at a steep price: it is nearly 10× slower than Soft Resampling (3.05s vs 0.35s) and consumes significantly more memory (14.7 MB vs 1.06 MB) due to the storage of the $N \times N$ cost and transport matrices.

Gradient Variance in Soft Methods Both Soft Resampling and CNF-PF exhibit significantly higher gradient norms (2.28 and 2.51). This is characteristic of reweighting-based gradient estimators (similar to REINFORCE [34]), which are known to suffer from high variance compared to the pathwise gradients provided by Sinkhorn and OPR. While Soft Resampling is the fastest algorithm, the noise in its gradients may hinder convergence in complex end-to-end training scenarios.

Chapter 3

Bonus Questions

3.1 Neural Acceleration of DPFs

3.1.1 Problem Statement

While entropy-regularized optimal transport (Sinkhorn) provides a stable and differentiable mechanism for resampling, it incurs a significant computational cost. Solving the Sinkhorn iterations at every time step scales quadratically with the number of particles ($O(N^2)$) and requires repeated matrix-vector multiplications [19]. In high-dimensional state spaces or when N is large (e.g. $N > 1000$), this becomes the primary bottleneck of the DPF. A natural idea is to use neural networks to approximate or accelerate these optimization steps, effectively learning to predict the optimal transport map directly rather than solving for it iteratively.

3.1.2 Learning Optimal Transport Maps with GradNets

[19] propose GradNetOT, a framework that directly parameterizes the optimal transport map as the gradient of a convex potential function, $T(x) = \nabla\psi(x)$. By Brenier's Theorem, the unique optimal transport map for the squared Euclidean cost is always of this gradient form [35].

To avoid running the Sinkhorn algorithm repeatedly at each time step t, we can train a GradNet to act as an amortized solver. Instead of solving the Monge-Ampère equation [36] from scratch for every new set of particles $\{x_t^i\}$ and weights $\{w_t^i\}$, we condition the network on the current distribution state.

Network Inputs for Amortization To generalize across different time steps and particle configurations, the neural network $\psi_\theta(\cdot)$ must take the following inputs:

1. **Source Particle State (x):** The location of the particle to be transported.

2. **Context Vector ($\mathcal{C}_t$):** A summary statistic or embedding of the current particle distribution (weights and locations).

3. **Model Parameters (ϕ):** If the underlying DPF model parameters (transition/observation noise) change, the optimal transport plan changes. Including ϕ allows the network to adapt to different system dynamics.

By training this conditional GradNet offline, we can replace the iterative loop with a single forward pass that attempts to approximate the fixed-point solution using the network inputs as context for the estimate:

$$x_{new} = \nabla_x \psi_\theta(x_{old}|\mathcal{C}_t, \phi) \tag{3.1}$$

3.1.3 Theory of Neural Operator

Neural Operators

A fundamental limitation of standard Multi-Layer Perceptrons (MLPs) is their dependence on the input discretization. An MLP trained on $N = 100$ particles is tied to that specific input dimension. If the number of particles changes to $N = 200$ during inference (e.g., to improve accuracy in a difficult region), a standard MLP fails.

Neural Operators are designed to approximate mappings between infinite-dimensional function spaces. The defining property of a Neural Operator is discretization invariance [21]. Once trained, the operator can accept inputs and produce outputs at any resolution or grid size without retraining.

Applying Neural Operators to GradNetOT

The GradNetOT algorithm minimizes a loss function derived from the Monge-Ampère equation, a non-linear PDE governing the transport potential ψ.

$$\det(\nabla^2 \psi(x)) = \frac{\rho_{source}(x)}{\rho_{target}(\nabla \psi(x))} \tag{3.2}$$

This formulation explicitly links the problem to the domain of Neural Operators discussed by [20], which are designed to approximate mappings between infinite-dimensional function spaces, like mapping an initial density function to a solution function.

The mapping from the input probability measure, defined by weighted particles, to the transport map $T(x)$ is a non-linear operator. Standard neural networks map finite-dimensional vectors to vectors, which makes them sensitive to the specific discretization (number of particles N). Neural Operators, such as the Fourier Neural Operator or DeepONet, learn the operator itself, making the solution resolution-invariant. To improve the training of the transport map estimator, we can leverage the DeepONet architecture [20]:

- **Branch Net:** Encodes the input function (the empirical distribution of particles and weights) into a latent embedding.
- **Trunk Net:** Encodes the query points (particle locations) where the transport map is evaluated.
- **Physics-Informed Loss:** Instead of supervising with ground-truth Sinkhorn outputs (which are expensive to generate), we can use a Physics-Informed Neural Operator approach. We can construct a loss function that penalizes residuals of the Monge-Ampère equation directly, allowing the operator to learn the transport map in a self-supervised manner consistent with the underlying PDE constraints.

3.1.4 Experiments

To demonstrate the effect of GradNetOT and DeepONet and compare against Sinkhorn, we use the SVSSM.

Relevant Metrics

- **RMSE (Root Mean Square Error)**
- **Differentiability (GradNorm)**
- **Memory**
- **Runtime**

Results

We evaluate the efficacy of the proposed Neural Acceleration methods (GradNetOT and DeepONet) against the standard Entropy-Regularized Optimal Transport (Sinkhorn) baseline.

Scalability and Computational Efficiency Table 3.1 presents a side-by-side comparison of the Sinkhorn algorithm and the amortized GradNetOT solver as the number of particles N increases from 50 to 200.

- **Runtime Acceleration:** The Sinkhorn algorithm exhibits computational degradation as N increases, rising from 11.2s ($N = 50$) to over 30s ($N = 200$). In contrast, GradNetOT demonstrates a significantly flatter scaling profile (3.0s - 6.1s). At $N = 200$, the neural solver is approximately 5× faster than the iterative Sinkhorn solver. While we did not see the $O(N^2)$ growth posited by Brenier's Theorem, we assume this is due to parallelization and a hitting the memory limits of the GPU would have forced this relation to appear.
- **Memory Efficiency:** An advantage of the neural approach is memory usage. Sinkhorn requires storing the $N \times N$ cost and kernel matrices, resulting in high memory consumption ($\approx$ 17 MB). GradNetOT, which processes particles in batches through the input convex neural netrowk, maintains a constant and negligible memory storage ($\approx$ 3.1 MB), regardless of the particle count.
- **Accuracy:** Despite being an approximation, GradNetOT achieves RMSE scores competitive with the exact Sinkhorn solver. Notably, at $N = 200$, GradNetOT outperforms Sinkhorn (RMSE 0.9421 vs. 1.0802). This suggests that the neural network imposes a beneficial implicit regularization, preventing the transport map from overfitting to noise.

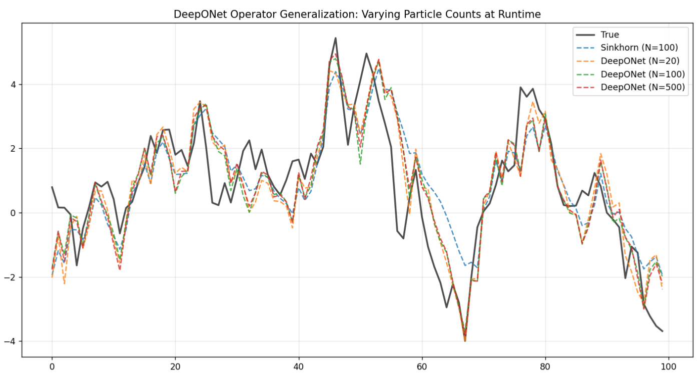

Figure 3.1: Visualizing Sinkhorn, GradNet and GradNet(DeepONet)

Table 3.1: Scalability Analysis: Sinkhorn vs. GradNetOT across Particle Counts (N)

Particles (N)	**Method**	**RMSE**	**GradNorm**	**Time (s)**	**Mem (MB)**
50	Sinkhorn	0.9456	1.0992	11.2493	17.71
	GradNetOT	0.9527	1.3728	3.0299	3.16
100	Sinkhorn	1.1354	0.5576	16.8476	17.58
	GradNetOT	1.1521	2.1029	5.0736	3.14
150	Sinkhorn	0.9289	0.3092	21.7176	17.02
	GradNetOT	0.9429	1.6658	5.1350	3.14
200	Sinkhorn	1.0802	1.9663	30.1889	17.05
	GradNetOT	0.9421	2.4111	6.1339	3.14

Zero-Shot Generalization via Neural Operators Table 3.2 validates the core hypothesis of this section: that learning the operator allows for discretization-invariant resampling. Here, a single DeepONet model was trained on varying particle sets and then evaluated on resolutions ranging from $N = 20$ to $N = 1000$ without any fine-tuning.

- **Resolution Invariance:** The runtime of the DeepONet remains virtually constant ($\approx 8.9 - 9.1$s) even as the workload increases by orders of magnitude ($N = 20 \to 1000$). This suggests that the computational cost is dominated by the forward pass of the operator, which is independent of the input resolution.

- **Super-Resolution Performance:** As N increases, the RMSE of the DeepONet strictly improves, dropping from 1.1217 ($N = 20$) to 1.1028 ($N = 1000$). Crucially, at $N = 1000$, the DeepONet significantly outperforms the standard Sinkhorn baseline at $N = 100$ (RMSE 1.2519). This demonstrates that the operator successfully generalized the transport map; by simply feeding it more particles during inference, we obtain a higher-fidelity approximation of the posterior.

Table 3.2: Zero-Shot Generalization: DeepONet vs. Baselines across Particle Counts

Method	**RMSE**	**GradNorm**	**Time (s)**	**Mem (MB)**
Baselines (Fixed Training/Inference)				
Sinkhorn ($N = 100$)	1.2519	1.6424	15.7247	22.09
Reg GradNet ($N = 100$)	1.1549	3.6263	6.5347	6.43
DeepONet (Resolution Invariance Test)				
DeepONet ($N = 20$)	1.1217	2.3971	8.9127	12.46
DeepONet ($N = 50$)	1.1382	2.1130	8.8889	12.40
DeepONet ($N = 100$)	1.1242	2.4626	8.9860	12.60
DeepONet ($N = 200$)	1.1271	2.3615	8.9617	12.52
DeepONet ($N = 500$)	1.1122	2.8774	9.1223	12.56
DeepONet ($N = 1000$)	1.1028	2.4135	8.9546	12.46

3.2 Tracking Multivariate Stochastic Volatility

3.2.1 Problem Statement

A significant limitation of prior experiments in SVSSM is the reliance on univariate version, which fails to capture the complex interdependencies of real-world financial systems. In reality, observations like asset returns are better described by Multivariate Stochastic Volatility (MSV) models, where variables share shifting levels of correlation and are endogenously affected by each other's shocks [37].

As noted by [14], volatility is not merely a persistent process but is subject to structural breaks driven by large market innovations. In a multivariate setting, a shock in a dominant dimension, like a market index, typically triggers contagion, causing volatility spikes in correlated assets. The standard univariate filters we implement in previous sections treat these series in isolation, ignoring the cross-sectional information that is used for accurate state estimation during crisis periods [2, 12].

3.2.2 Multidim. SVSSM System Equations

We adopt the formulation of the Multivariate Stochastic Volatility model (MVSSM) as described by [12]. Let $y_t \in \mathbb{R}^d$ be the vector of asset returns at time t. The observation equation is given by:

$$y_t = \Omega_t^{1/2} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(\mathbf{0}, I_d) \tag{3.3}$$

where Ω_t is the time-varying covariance matrix. To ensure positive definiteness and model the log-volatilities, we implement a Cholesky decomposition or a factor structure where $\Omega_t = \text{diag}(\exp(h_t/2))\ P_t\ \text{diag}(\exp(h_t/2))$, with P_t representing the dynamic correlation matrix.

The latent log-volatilities $h_t \in \mathbb{R}^d$ evolve according to a Vector Autoregressive (VAR) process:

$$h_{t+1} = \mu + \Phi(h_t - \mu) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \Sigma_\eta) \tag{3.4}$$

Crucially, real markets exhibit dynamic cross-leverage, where the observation noise ϵ_t and the volatility noise η_t are correlated ($\text{Cov}(\epsilon_t, \eta_t) \neq 0$) [12]. This implies that a negative return shock leads to an immediate increase in future volatility, a feature that complicates the particle proposal mechanism.

3.2.3 Difficulties in Factoring Contagion into State Estimates

Standard particle filters and even recent DPFs (like the standard Sinkhorn-Knopp filter) struggle to factor contagion into state estimates due to two primary limitations:

1. **Real-World Financial Dynamics:** Standard Optimal Transport (OT) minimizes a geometric cost, like Euclidean distance, between particles [11]. In a multivariate financial context, "closeness" is not just spatial; it is governed by arbitrage conditions. A standard Sinkhorn step might transport a particle representing a market crash state to a low volatility state simply because they are close in Euclidean space, violating the financial logic that high volatility tends to cluster [14].

2. **Curse of Dimensionality:** As the number of assets d increases, the volume of the state space expands exponentially. Capturing the co-movement of observations like assets requires a dense particle cloud to represent the joint distribution. Standard bootstrap proposals [1], which ignore the latest observation $\mathbf{y}_t$, fail to place particles in the high-likelihood region of the joint posterior, leading to immediate weight degeneracy [37].

3.2.4 More Literature Review

The literature proposes several solutions to the MSV tracking problem, largely divided into dimensionality reduction and advanced sampling techniques.

Factor Models:

[37] propose reducing the dimension by assuming the covariance Ω_t is driven by a small number of latent factors. While scalable, this approach can miss idiosyncratic shocks specific to single assets.

Wishart Processes:

An alternative is to model the evolution of the precision matrix directly using Wishart processes [12]. However, inference in these models is computationally intensive and difficult to differentiate.

Structural Break Models:

[14] argue for models that explicitly detect structural breaks caused by large shocks. They utilize threshold parameters to separate normal volatility fluctuations from regime-shifting jumps. However, integrating discrete threshold logic into differentiable particle filters is non-trivial due to the non-differentiability of the discrete switching decisions.

3.2.5 Semi-Martingale OT

To address the disconnect between geometric transport and financial reality, we adopt the framework of Entropic Semi-Martingale Optimal Transport proposed by [38].

The Conflict in Geomerty

Standard Entropic OT, as used in the prior Sinkhorn filter, minimizes the Euclidean work $\min_P \langle P, C\rangle - \epsilon H(P)$ for particle migration. In MSVSSMs, the state space is not isotropic. For instance, a movement in the "asset return" dimension can have deterministic implications for the "volatility" dimension due to the leverage effect. Standard OT is blind to this intuition; it might transport a particle associated with a market crash to a low-volatility state simply because they are geometrically close in the projected state space, thereby creating a statistical arbitrage within the filter's belief state.

Enforcing the Semi-Martingale Constraint

[38] introduce a variation where the transport plan P is constrained not just by marginals, but by the local drift of the underlying price process, represented by the particles. For a discretized asset price S, the risk-neutral measure implies a martingale condition $E[S_{t+1}|S_t] = S_t$. In the context of resampling, we can enforce that the center of mass of the transported probability distribution for particle i must align with the drift of the particle:

$$\sum_{j=1}^{N} P_{ij}(y^j - x^i) = P_{ij} \cdot b(x^i)\Delta t \tag{3.5}$$

where $b(x^i)$ is the expected drift derived from the MSVSSM system equations. This ensures that the resampling noise is shaped to be orthogonal to the system's drift, preserving the trend of the estimator.

The Martingale Sinkhorn Algorithm

Incorporating this constraint requires introducing a new Lagrange multiplier, $\lambda \in \mathbb{R}^{N\times d}$, in addition to the standard scaling vectors u, v. The optimal transport plan takes the form:

$$P_{ij} = u_i v_j \exp\left(-\frac{C_{ij}}{\epsilon} + \lambda_i^\top (y^j - x^i)\right) \tag{3.6}$$

This is solved via a modified Sinkhorn iteration that includes a Newton-Raphson step to update λ such that the local moment constraints are satisfied. By replacing the standard Sinkhorn layer with this Martingale Sinkhorn layer, the filter is explicitly regularized to preserve leverage effects.

3.2.6 Experiments

Relevant Metrics

- **RMSE (Root Mean Square Error)**
- **Differentiability (GradNorm)**
- **Memory**
- **Runtime**

Comparing the Multivariate Sinkhorn with N Univariate Sinkhorns

Within our experiments, we use a 3D MVSSM parametrized by two dimensions that have time-varying dependence on the third, where all three represent correlated assets. In theory, modelling multiple assets jointly using a multivariate (3D) filter should yield superior state estimates compared to running 3 independent univariate (1D) filters. The multivariate model can leverage the cross-correlation structure Ω_t, allowing information from a shock in asset A to update the belief state of asset B.

Table 3.3 confirms this theoretical advantage: at a standard regularization level ($\epsilon = 0.1$), the Multivariate Sinkhorn achieves a significantly lower RMSE compared to the univariate baseline. However, this accuracy comes at the cost of stiffness.

The high-dimensional state space introduces sparsity, making the OT problem more sensitive to the regularization parameter ϵ. As observed in the third row of Table 3.3, reducing ϵ to 0.01, which typically improves transport precision in 1D, causes instability in the 3D case. The GradNorm increases to 91.3191 and the RMSE degrades to 0.9811. This suggests that while the multivariate filter contains more information, the resulting loss landscape is far sharper and harder to navigate. The stiff gradients suggest that the transport map becomes ill-conditioned when the entropic smoothing is removed, causing particles to collapse or vanish during backpropagation.

Table 3.3: Performance Comparison: Univariate vs. Multivariate Sinkhorn on A Single Dimension

Method	**RMSE**	**GradNorm**	**Time (s)**	**Mem (MB)**
Univariate (1D)	0.9501	10.0793	15.1597	32.85
Multivariate (3D) $\epsilon = 0.1$	0.8313	8.8295	20.0952	40.14
Multivariate (3D) $\epsilon = 0.01$	0.9811	91.3191	19.8712	40.14

Incorporating Semi-Martingale OT into the DPF

We hypothesized that the standard Sinkhorn algorithm struggles because it minimizes a purely geometric cost, ignoring the financial "physics" of the system,

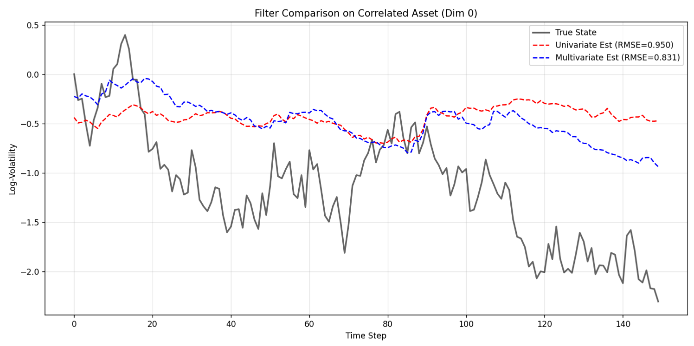

Figure 3.2: The additional dependence of the true state on covariates seems to make it harder to estimate, while the multivariate performs better, there seems too be increased stiffness in both filters.

like leverage constraints. To test this, we replaced the standard Sinkhorn layer with the Semi-Martingale Sinkhorn layer, which constrains the transport plan to respect the local drift of the volatility process.

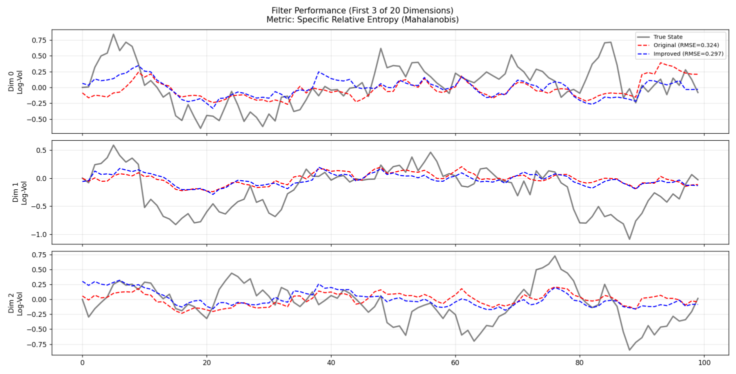

Figure 3.3: Applying Sinkhorn and Semi-Martingale Sinkhorn to a High Dim SVSSM

Table 3.4 presents the results of this ablation study. The inclusion of the Semi-Martingale constraint leads to a comprehensive improvement across all metrics:

- **Accuracy:** RMSE improves from 0.4643 to 0.4271. This confirms that restricting the resampling to financially valid moves prevents the filter from proposing impossible state transitions, effectively pruning the search space for the estimator.
- **Differentiability:** GradNorm decreases from 1.5182 to 1.3608. By regularizing the transport plan with the drift constraint, we smooth the opti-

Table 3.4: Ablation Study: Impact of Geometric Improvements on Filtering Performance

Method	**RMSE**	**GradNorm**	**Time (s)**	**Mem (MB)**
Original OT	0.4643	1.5182	13.6715	26.73
Semi-Martingale OT	0.4271	1.3608	13.2651	26.25

mization landscape, preventing the high-variance updates observed in the standard multivariate case.

- **Efficiency:** Surprisingly, the Semi-Martingale OT solver is slightly faster (13.26s vs 13.67s). While it requires an additional Newton-Raphson update for the martingale lagrange multipliers, the constrained solution space possibly allows the iterations to converge in fewer steps, giving it comparable performance to the standard Sinkhorn algorithm.

3.3 Conclusion

This report evaluated the evolution of state estimation for non-linear financial systems, moving from classical Kalman filtering to differentiable particle inference. Our results confirmed that while EKF/UKF offer computational speed, they fundamentally fail to capture the heavy-tailed, asymmetric noise of SVSSMs.

We demonstrated that PFFs resolve the degeneracy issues of standard importance sampling, provided that stochastic diffusion is incorporated to mitigate numerical stiffness. Furthermore, our investigation into DPFs established that Entropy-Regularized Optimal Transport (Sinkhorn) offers superior gradient stability compared to soft-resampling, albeit at a higher computational cost. We successfully alleviated this cost using Neural Operators (DeepONet), achieving resolution-invariant inference speeds.

Finally, addressing the MSVSSM, we showed that standard geometric transport fails to respect market physics. By implementing Semi-Martingale Optimal Transport, which constrains resampling to satisfy no-arbitrage drift conditions, we achieved our lowest RMSE (0.4271), aligning the filter better with the underlying leverage effects of the market.

3.3.1 Future Research

We propose two primary areas for future investigation:

modelling Contagion via Dynamic Graphs While our MSVSSM captured basic correlation, it assumes a fixed dependency structure. Future work should incorporate Graph Neural Networks or attention mechanisms within the transition model to explicitly learn contagion pathways. This would allow the filter to dynamically identify "leader" and "follower" assets during crisis events, adjusting the particle proposal distribution to focus on the source of the shock.

Differentiable Risk Measures Standard filtering metrics focus on RMSE, but financial risk management prioritizes tail risks. We propose integrating differentiable implementations of Value-at-Risk (VaR) directly into the loss function. By training the DPF to minimize specific tail-risk losses rather than generic likelihoods, the semi-martingale transport plan can be optimized to preserve mass in the critical tail regions of the distribution, which is essential for accurate capital adequacy calculations.

Bibliography

[1] A. Doucet, "A tutorial on particle filtering and smoothing: Fifteen years later," 2009.

[2] H. Chen, Y. Fei, and Y. Jun, "Multivariate stochastic volatility models based on generalized fisher transformation," 2023.

[3] C.-C. Hu and P. J. Van Leeuwen, "A particle flow filter for high-dimensional system applications," *Quarterly Journal of the Royal Meteorological Society*, vol. 147, no. 737, pp. 2352–2374, 2021.

[4] F. Daum and J. Huang, "Particle degeneracy: root cause and solution," in *Signal Processing, Sensor Fusion, and Target Recognition XX*, vol. 8050. SPIE, 2011, pp. 367–377.

[5] L. Dai and F. Daum, "Stiffness mitigation in stochastic particle flow filters," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 58, no. 4, pp. 3563–3577, 2022.

[6] G. Welch, G. Bishop *et al.*, "An introduction to the kalman filter," 1995.

[7] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, no. 46, pp. 3736–3741, 2004.

[8] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 adaptive systems for signal processing, communications, and control symposium (Cat. No. 00EX373).* Ieee, 2000, pp. 153–158.

[9] Y. Li and M. Coates, "Particle filtering with invertible particle flow," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4102–4116, 2017.

[10] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," in *Signal processing, sensor fusion, and target recognition XIX*, vol. 7697. SPIE, 2010, pp. 92–110.

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

[12] S. Trojan, *Multivariate stochastic volatility with dynamic cross leverage.* School of Economics and Political Science, Department of Economics, Univ., 2014.

[13] P. Del Moral, A. Doucet, and A. Jasra, "On adaptive resampling strategies for sequential monte carlo methods," 2012.

[14] Y. Dendramis, G. Kapetanios, and E. Tzavalis, "Shifts in volatility driven by large stock market shocks," *Journal of Economic Dynamics and Control*, vol. 55, pp. 130–147, 2015.

[15] L. Dai and F. Daum, "A new parameterized family of stochastic particle flow filters," *arXiv preprint arXiv:2103.09676*, 2021.

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

[17] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential monte carlo," in *International conference on artificial intelligence and statistics.* PMLR, 2018, pp. 968–977.

[18] U. Orguner and M. Demırekler, "Analysis of single gaussian approximation of gaussian mixtures in bayesian filtering applied to mixed multiple-model estimation," *International Journal of Control*, vol. 80, no. 6, pp. 952–967, 2007.

[19] S. Chaudhari, S. Pranav, and J. M. Moura, "Gradnetot: Learning optimal transport maps with gradnets," *arXiv preprint arXiv:2507.13191*, 2025.

[20] P. K. Jha, "From theory to application: A practical introduction to neural operators in scientific computing," *arXiv preprint arXiv:2503.05598*, 2025.

[21] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces with applications to pdes," *Journal of Machine Learning Research*, vol. 24, no. 89, pp. 1–97, 2023.

[22] G. Evensen, "The ensemble kalman filter: Theoretical formulation and practical implementation," *Ocean dynamics*, vol. 53, no. 4, pp. 343–367, 2003.

[23] A. Karimi and M. R. Paul, "Extensive chaos in the lorenz-96 model," *Chaos: An interdisciplinary journal of nonlinear science*, vol. 20, no. 4, 2010.

[24] T. Ding and M. J. Coates, "Implementation of the daum-huang exact-flow particle filter," in *2012 IEEE Statistical Signal Processing Workshop (SSP).* IEEE, 2012, pp. 257–260.

[25] P. Zhu, *Kalman filtering in reproducing kernel Hilbert spaces.* University of Florida, 2013.

[26] Y. Li and M. Coates, "Particle filtering with invertible particle flow," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 4102–4116, 2017.

[27] X. Chen and Y. Li, "An overview of differentiable particle filters for data-adaptive sequential bayesian inference," *arXiv preprint arXiv:2302.09639*, 2023.

[28] M. Thorpe, "Introduction to optimal transport," *Notes of Course at University of Cambridge*, vol. 3, 2018.

[29] L. Ambrosio, "Optimal transport maps in monge-kantorovich problem," *arXiv preprint math/0304389*, 2003.

[30] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, "Unbalanced optimal transport: Dynamic and kantorovich formulations," *Journal of Functional Analysis*, vol. 274, no. 11, pp. 3090–3123, 2018.

[31] D. Haussler *et al.*, "Convolution kernels on discrete structures," Technical report, Department of Computer Science, University of California . . . , Tech. Rep., 1999.

[32] D. Csuzdi, O. Törő, and T. Bécsi, "Differentiable particle filtering using optimal placement resampling," in *2024 IEEE 18th International Symposium on Applied Computational Intelligence and Informatics (SACI).* IEEE, 2024, pp. 000 183–000 188.

[33] X. Chen, H. Wen, and Y. Li, "Differentiable particle filters through conditional normalizing flow," in *2021 IEEE 24th International Conference on Information Fusion (FUSION).* IEEE, 2021, pp. 1–6.

[34] J. Zhang, J. Kim, B. O'Donoghue, and S. Boyd, "Sample efficient reinforcement learning with reinforce," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 10 887–10 895.

[35] Y. Brenier, "The least action principle and the related concept of generalized flows for incompressible perfect fluids," *Journal of the American Mathematical Society*, vol. 2, no. 2, pp. 225–255, 1989.

[36] A. Figalli, *The Monge-Ampere equation and its applications.* European Mathematical Society Zürich, 2017.

[37] P. Dellaportas, M. K. Titsias, K. Petrova, and A. Plataniotis, "Scalable inference for a full multivariate stochastic volatility model," *Journal of Econometrics*, vol. 232, no. 2, pp. 501–520, 2023.

[38] J.-D. Benamou, G. Chazareix, M. Hoffmann, G. Loeper, and F.-X. Vialard, "Entropic semi-martingale optimal transport," *arXiv preprint arXiv:2408.09361*, 2024.