Classical Filters to Particle Flow Filters

Amresh Verma

February 2, 2026

Abstract

This report presents a comprehensive study of sequential state estimation methods across linear, nonlinear, and high-dimensional regimes. We implement and analyze the Kalman filter (with Riccati and Joseph updates), EKF, UKF, and particle filter on canonical and custom state-space models, diagnosing numerical stability, linearization failure, sigma-point collapse, and particle degeneracy using RMSE, NEES/NIS, ESS, and conditioning metrics. Building on these foundations, we replicate and extend recent particle-flow and kernel-embedded flow methods (Li, 2017; Hu, 2021), showing that invertible particle flows yield large accuracy gains in nonlinear multi-target tracking while matrix-valued kernels prevent marginal collapse and stabilize assimilation in high dimensions. Results highlight a practical accuracy/efficiency tradeoff and position transport-based proposals as effective tools for challenging filtering problems.

1 Introduction

Scientific problems often require estimation of the hidden state of a system that changes continuously over time, using a sequence of noisy measurements made on the system. To model time-series in the state-space approach we use a state vector of the system that contains all relevant information required to describe the system under investigation, e.g., in tracking problems, this information is related to the kinematic characteristics of the target, in an econometrics problem, it could be related to monetary flow, interest rates, inflation, etc. In most cases, it is not possible to directly observe the state vector. Instead, a related set of variables are usually observed. This set is referred to as the *measurements*. The relationship between *measurements* and *state* can either be linear or non-linear. In real world, measurements are contaminated by random sources of noise.

The state vector is usually (but not necessarily) of higher dimension than the measurements vector. The state-space approach is convenient for handling multivariate data and nonlinear/non-Gaussian processes. It consistently provides a significant advantage over traditional time-series techniques for these problems [1]. Many varied examples illustrating the application of nonlinear/nonGaussian state space models are given in [2].

Analysis and inference about any dynamical system require two models: 1. a model describing the time evolution of state, called the system model, and 2. a model that relates measurements to the state, called the measurement model. This state-space formulation requires updating the information as new measurements are made available, and this creates an ideal scenario for employing a Bayesian approach. This provides a rigorous general framework for dynamic state estimation problems. The Bayesian method aims to construct a posterior probability distribution function, sometimes simply referred to as *posterior*, of the state based on all available information, including the set of available measurements. In many problems, measurements are taken recursively, and we must make an estimate with each measurement, which requires a recursive filter. This approach allows us to process the data sequentially rather than as a batch, so it is not necessary to store the complete data. Such a filter works in two essential stages: prediction and update. In the prediction stage, the system model predicts the state from one time step to the next. During this stage, unknown disturbances and noise usually deform the state probability distribution function. The update operation makes use of the latest measurement to modify the prediction probability. This is achieved using the Bayes' theorem, which serves as the mechanism for updating our current knowledge in the presence of additional information derived from new data.

This report is organized as follows. Section 2 provides a brief review of general state-space models (SSMs), followed by subsections detailing the theoretical foundations of all filtering methods considered. Section 3 presents the experimental results and addresses the project questions. Finally, Section 4 provides a conclusion.

2 State-Space Model and Filters

A wide class of dynamical systems can be described using a discrete-time state-space model (SSM), defined by the equations

$$\mathbf{x}_k = f_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}), \tag{1}$$
$$\mathbf{y}_k = h_k(\mathbf{x}_k, \mathbf{w}_k), \tag{2}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_x}$ denotes the state vector at time k, $\mathbf{y}_k \in \mathbb{R}^{n_y}$ is the measurement vector, $f_k(\cdot)$ and $h_k(\cdot)$ are known state transition and measurement functions, respectively, and $\mathbf{v}_{k-1} \in \mathbb{R}^{n_v}$ and $\mathbf{w}_k \in \mathbb{R}^{n_w}$ represent the process and measurement noise sequences. The noise processes are typically assumed to be mutually independent and independent and identically distributed (i.i.d.).

The objective of Bayesian filtering is to recursively infer the state $\mathbf{x}_k$ from the sequence of measurements $\mathbf{y}_{1:k} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ by constructing the posterior probability density function (pdf) $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$. It is assumed that the initial state pdf $p(\mathbf{x}_0)$ is known.

Under the Markov assumption, the posterior density can be computed recursively using a two-step procedure consisting of prediction and update. Given the posterior density at time $k-1$, the prediction step evaluates the prior density at time k using the Chapman–Kolmogorov equation

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{y}_{1:k-1})\, d\mathbf{x}_{k-1}. \tag{3}$$

The update step incorporates the new measurement $\mathbf{y}_k$ through Bayes' rule,

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) = \frac{p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})}. \tag{4}$$

where the normalizing constant

$$p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) = \int p(\mathbf{y}_k \mid \mathbf{x}_k)\, p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})\, d\mathbf{x}_k \tag{5}$$

depends on the likelihood function defined by the measurement model (2) and the known statistics of $\mathbf{w}_k$. In the update step (4), the measurement is used to modify the prior density to obtain the posterior density of the current state.

The recurrence relations (3) and (4) form the basis of the optimal Bayesian solution. This recursive propagation of the posterior density is, however, only a conceptual solution in that, in general, it cannot be determined analytically. Closed-form solutions exist in a restrictive set of cases, such as the linear Gaussian model where the Kalman filter applies. When the analytic solution is intractable, various approximations to the optimal Bayesian solution can be employed, including the extended Kalman filter (EKF), unscented Kalman filter (UKF), particle filters (PF), and other variants such as EDH, LEDH, invertible PFPF, and kernel-based particle flow filters, which are discussed in the next section.

2.1 Kalman Filter

The Kalman filter assumes that the posterior density at every time step is Gaussian and is therefore completely characterized by its mean and covariance [3]. If the prior density is Gaussian, posterior density remains Gaussian, provided that the following assumptions hold: 1. The process noise $\mathbf{v}_{k-1}$ and measurement noise $\mathbf{w}_k$ are drawn from Gaussian distributions, 2. the state transition model is linear in the state and process noise, 3. the measurement model is linear in the state and measurement noise. Under these assumptions, the state-space model in (1)–(2) can be rewritten as

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k \mathbf{v}_{k-1}, \tag{6}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k, \tag{7}$$

where $\mathbf{F}_k$ and $\mathbf{H}_k$ are known system and measurement matrices, and $\mathbf{G}_k$ defines how the process noise enters the state equation. The process and measurement noise covariances are given by $\mathbf{v}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$, where the noise processes are assumed to be zero-mean. The system and measurement matrices $\mathbf{F}_k$ and $\mathbf{H}_k$, as well as the noise matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$, can be time dependent. Using the Bayesian prediction and update relations in (3) and (4), the Kalman filter algorithm can be expressed as the following recursive equations:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1}, \tag{8}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^\top + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^\top, \tag{9}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1}\right), \tag{10}$$

where $\hat{\mathbf{x}}_{k|k-1}$ and $\hat{\mathbf{x}}_{k|k}$ denote the predicted and filtered state estimates, respectively, $\mathbf{P}_{k|k-1}$ is the predicted error covariance, and $\mathbf{K}_k$ is the Kalman gain defined as:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1}, \tag{11}$$

and the posterior error covariance is updated according to the discrete-time Riccati equation

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1}. \tag{12}$$

For improved numerical stability, the posterior covariance update in (12) is often implemented using the equivalent Joseph stabilized form

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\, \mathbf{P}_{k|k-1} \,(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^{\top} + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^{\top}. \tag{13}$$

The Kalman filter provides the optimal Bayesian solution, but only under the assumptions of linearity and Gaussian noise, which are often violated in practical applications. These limitations motivate the development of approximate filters for nonlinear and non-Gaussian systems, which are discussed in the following sections.

2.2 Extended Kalman Filter

If the state-space equations (1) and (2) cannot be modeled in the linear forms (6) and (7) due to nonlinear state transition or measurement functions, a local linearization of the model may be sufficient for an accurate description of the system dynamics [4]. The extended Kalman filter (EKF) is based on this approximation. The nonlinear functions are linearized about the current state estimate using a first-order Taylor series expansion, and the resulting posterior density is approximated as Gaussian:

$$\mathbf{x}_k \approx \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{v}_{k-1}, \tag{14}$$

$$\mathbf{y}_k \approx \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k, \tag{15}$$

here $\mathbf{F}_k$ and $\mathbf{H}_k$ denote the Jacobian matrices of the state transition and measurement functions, respectively. The Jacobians are defined using f_k, h_k from equations (1) and (2) as

$$\mathbf{F}_k = \left. \frac{\partial f_k(\mathbf{x}, \mathbf{v})}{\partial \mathbf{x}} \right|_{\mathbf{x} = \hat{\mathbf{x}}_{k-1|k-1},\, \mathbf{v} = \mathbf{0}}, \tag{16}$$

$$\mathbf{H}_k = \left. \frac{\partial h_k(\mathbf{x}, \mathbf{w})}{\partial \mathbf{x}} \right|_{\mathbf{x} = \hat{\mathbf{x}}_{k|k-1},\, \mathbf{w} = \mathbf{0}}, \tag{17}$$

Apart from the local linearization of the state transition and measurement models, the EKF follows the same recursive prediction and update structure (from (8) to (13)) as the Kalman filter at each time step.

2.3 Unscented Kalman Filter

The UKF is also an attempt to address nonlinear state-space models like the EKF. However, instead of linearizing $f(\cdot)$ and $h(\cdot)$ via Jacobians, the UKF uses the *unscented transform*, which propagates the mean and covariance through nonlinear functions accurately up to the third order for Gaussian distributions [5]. This is achieved using the sigma points.

For an n-dimensional state, $(2n+1)$ sigma points are generated around the current mean $\hat{\mathbf{x}}$ to capture the covariance $\mathbf{P}$:

$$\chi_0 = \hat{\mathbf{x}}, \tag{18}$$

$$\chi_i = \hat{\mathbf{x}} + \left[\sqrt{(n+\lambda)\mathbf{P}}\right]_i, \quad i = 1, \ldots, n, \tag{19}$$

$$\chi_{i+n} = \hat{\mathbf{x}} - \left[\sqrt{(n+\lambda)\mathbf{P}}\right]_i, \tag{20}$$

where $\lambda = \alpha^2(n+\kappa) - n$ and α, β, κ are tuning parameters. The matrix square root is typically computed via Cholesky decomposition. Each sigma point has associated mean and covariance weights:

$$W_0^m = \frac{\lambda}{n+\lambda}, \qquad W_0^c = W_0^m + (1 - \alpha^2 + \beta), \tag{21}$$

$$W_i = \frac{1}{2(n+\lambda)}, \qquad i = 1, \ldots, 2n. \tag{22}$$

Prediction: Propagate sigma points through the motion model:

$$\chi_i^- = f(\chi_i, \mathbf{u}_k), \tag{23}$$

$$\hat{\mathbf{x}}_{k|k-1} = \sum_i W_i^m \chi_i^-, \tag{24}$$

$$\mathbf{P}_{k|k-1} = \sum_i W_i^c (\chi_i^- - \hat{\mathbf{x}}_{k|k-1})(\chi_i^- - \hat{\mathbf{x}}_{k|k-1})^\top + \mathbf{Q}. \tag{25}$$

Update: Transform sigma points through the measurement model:

$$\mathbf{z}_i = h(\chi_i^-), \tag{26}$$

$$\hat{\mathbf{z}}_k = \sum_i W_i^m \mathbf{z}_i, \tag{27}$$

$$\mathbf{S}_k = \sum_i W_i^c (\mathbf{z}_i - \hat{\mathbf{z}}_k)(\mathbf{z}_i - \hat{\mathbf{z}}_k)^\top + \mathbf{R}, \tag{28}$$

$$\mathbf{P}_{xz} = \sum_i W_i^c (\chi_i^- - \hat{\mathbf{x}}_{k|k-1})(\mathbf{z}_i - \hat{\mathbf{z}}_k)^\top, \tag{29}$$

$$\mathbf{K}_k = \mathbf{P}_{xz}\mathbf{S}_k^{-1}, \tag{30}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k), \tag{31}$$

$$\mathbf{P}_k = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \tag{32}$$

There are multiple advantages of the UKF over the EKF. The UKF is a *derivative-free* EKF, avoiding the need for Jacobians, and it often provides more accurate estimates than the EKF for highly nonlinear systems.

2.4 Particle Filter

The sequential importance sampling (SIS) algorithm is a Monte Carlo method forming the basis for most sequential MC filters developed over the the time. The particle filter (PF) is a sequential Monte Carlo method that provides an approximate solution to the Bayesian filtering problem for general nonlinear and non-Gaussian state-space models [6]. The PF approximates the posterior distribution $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ using a set of N weighted particles $\{\mathbf{x}_k^{(i)}, w_k^{(i)}\}_{i=1}^N$:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta(\mathbf{x}_k - \mathbf{x}_k^{(i)}), \tag{33}$$

where $\delta(\cdot)$ denotes the Dirac delta function. The algorithm proceeds recursively in two steps:

Prediction: Each particle is propagated through the system dynamics:

$$\mathbf{x}_k^{(i)} \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}^{(i)}). \tag{34}$$

Update: The particle weights are updated based on the likelihood of the measurement:

$$w_k^{(i)} \propto w_{k-1}^{(i)} \frac{p(\mathbf{y}_k \mid \mathbf{x}_k^{(i)}) p(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)})}{q(\mathbf{x}_k^{(i)} \mid \mathbf{x}_{k-1}^{(i)}, \mathbf{y}_k)}, \qquad \sum_{i=1}^{N} w_k^{(i)} = 1. \tag{35}$$

Here, $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ is the motion model, $p(\mathbf{z}_k \mid \mathbf{x}_k)$ is the likelihood of measurement, and $q(\cdot)$ is the *importance density*. To mitigate particle degeneracy, a resampling step is typically applied, in which particles with small weights are discarded and particles with large weights are replicated according to a chosen resampling scheme [7].

2.5 Daum-Huang Flow

In Daum–Huang Particle Flow, the standard Sequential Importance Sampling (SIS) update with weights and resampling is replaced by a deterministic flow of particles from the prior to the posterior [8]. It constructs an ordinary differential equation (ODE) that transports particles in pseudo-time such that the evolving particle density exactly matches the conditional posterior density, without needing weights or resampling.

EDH: The flow trajectory in the exact Daum-Huang (EDH) filter is:

$$\frac{dx_\lambda^i}{d\lambda} = A(\lambda) x_\lambda^i + b(\lambda), \tag{36}$$

with [9]

$$\begin{aligned} A(\lambda) &= -\tfrac{1}{2} P H^\top (\lambda H P H^\top + R)^{-1} H \\ b &= (I + 2\lambda A)\Big[(I + \lambda A) P H^\top R^{-1} (z - e) + A\bar{x}_0\Big], \quad e = h(\bar{x}_\lambda, 0) - H\bar{x}_\lambda \end{aligned}$$

For nonlinear observation models, a linearization of the model is performed at the mean of the intermediate distribution, $\bar{x}_\lambda$, to construct $H(\lambda)$:

$$H(\lambda) = \left.\frac{\partial h(x, 0)}{\partial x}\right|_{x=\bar{x}_\lambda}. \tag{37}$$

This Jacobian matrix is evaluated at $\bar{x}_\lambda$.

LEDH: In this method of localized exact Daum and Huang filter (LEDH), we linearize the system and update the drift term for each individual particle:

$$\frac{dx^i_\lambda}{d\lambda} = A^i(\lambda)x^i_\lambda + b^i(\lambda), \tag{38}$$

here $A^i(\lambda)$ and $b^i(\lambda)$ are of the same form as above, but for each particle we construct an $H^i(\lambda)$ as

$$H^i(\lambda) = \left.\frac{\partial h(x,0)}{\partial x}\right|_{x=\bar{x}^i_\lambda}, \quad e^i(\lambda) = h(\bar{x}^i_\lambda, 0) - H^i\bar{x}^i_\lambda \tag{39}$$

2.6 Particle Flow Particle Filter

Li and Coates [10] combine deterministic particle flow with particle filtering by incorporating importance weights and resampling. This approach retains the advantages of particle flow in transporting particles toward high-likelihood regions while allowing the method to handle non-Gaussian and multimodal posteriors that arise in highly nonlinear systems. In deterministic particle flow filters such as EDH and LEDH, the Bayesian update is implemented via a homotopy parameter $\lambda \in [0,1]$ that continuously transforms the prior $\pi_0(x) = p(x)$ to the posterior $\pi_1(x) \propto p(x)p(z \mid x)$. A linearized flow

$$\frac{dx(\lambda)}{d\lambda} = A(\lambda)x(\lambda) + b(\lambda) \tag{40}$$

ensures that the evolving density π_λ approximately satisfies the continuity (Liouville) equation. Discretizing λ into steps ε_j yields an affine map

$$x_{j+1} = M_j x_j + c_j, \quad M_j = I + \varepsilon_j A_j, \tag{41}$$

so that the full flow from prior to posterior is

$$\eta_1 = T(\eta_0) = T_{N_\lambda} \circ \cdots \circ T_1(\eta_0). \tag{42}$$

Here, η_0 denote a particle drawn from the prior, $\eta_0 \sim p(x)$, and $\eta_1$its transformed state after applying the full particle flow T.

By the change-of-variables formula, the induced density p_{η_1} involves the Jacobian determinant:

$$p_{\eta_1}(\eta_1) = p_{\eta_0}(\eta_0)\left|\det\frac{\partial\eta_1}{\partial\eta_0}\right|^{-1}, \quad \frac{\partial\eta_1}{\partial\eta_0} = \prod_{j=1}^{N_\lambda} M_j. \tag{43}$$

Importance weights for each particle are then

$$w \propto \frac{p(z \mid \eta_1)\,p(\eta_1)}{p(\eta_0)}\left|\det\frac{\partial\eta_1}{\partial\eta_0}\right|, \tag{44}$$

where the likelihood term $p(z \mid \eta_1)$, prior correction $p(\eta_1)/p(\eta_0)$, and Jacobian determinant together account for the flow's effect on particle density. In EDH, A_j is computed once using the ensemble mean, while in LEDH a particle-specific $A_{i,j}$ is used for local adaptation, requiring a determinant per particle. In this way, the particle-flow particle filter (PFPF) extends standard EDH/LEDH methods to non-Gaussian and multimodal inference tasks.

2.7 Kernel Particle Flow Filter (KPFF)

The kernel particle flow filter (KPFF) proposed by Hu and van Leeuwen [11] embeds the particle flow in a reproducing kernel Hilbert space (RKHS) to transport particles from the prior to the posterior while mitigating weight degeneracy. In this framework, the continuous flow for particle i is defined as

$$\frac{d\eta^{(i)}(\lambda)}{d\lambda} = \frac{1}{N} \sum_{j=1}^{N} \left[k(\eta^{(i)}, \eta^{(j)}) \nabla_{\eta^{(j)}} \log p(z \mid \eta^{(j)}) + \nabla_{\eta^{(j)}} k(\eta^{(i)}, \eta^{(j)}) \right], \tag{45}$$

where $k(\cdot, \cdot)$ is a positive-definite kernel, N is the ensemble size, and $\lambda \in [0, 1]$ is the homotopy parameter. The first term represents the weighted average of log-posterior gradients, while the second term accounts for the divergence of the kernel, ensuring the flow preserves particle density.

In the limit of an infinite ensemble ($N \to \infty$), the flow becomes independent of the kernel choice. For finite ensembles, however, using a *scalar kernel*—the same kernel for all state dimensions—can lead to variance collapse in high-dimensional, sparsely observed systems. To mitigate this, a *matrix-valued kernel* is introduced:

$$k(\eta^{(i)}, \eta^{(j)}) \in \mathbb{R}^{d \times d}, \quad \frac{d\eta^{(i)}}{d\lambda} = \frac{1}{N} \sum_{j=1}^{N} \left[\mathbf{K}(\eta^{(i)}, \eta^{(j)}) \nabla_{\eta^{(j)}} \log p(z \mid \eta^{(j)}) + \nabla_{\eta^{(j)}} \mathbf{K}(\eta^{(i)}, \eta^{(j)}) \right], \tag{46}$$

where $\mathbf{K}(\cdot, \cdot)$ is a $d \times d$ positive-definite kernel matrix, allowing different state components to adapt individually to the flow. This prevents collapse of marginal distributions and better preserves posterior uncertainty across dimensions.

Importance weights can still be applied optionally, similar to standard particle flow filters:

$$w^{(i)} \propto \frac{p(z \mid \eta_1^{(i)})\, p(\eta_1^{(i)})}{p_{\eta_1}(\eta_1^{(i)})}, \tag{47}$$

where the denominator accounts for the change of variables under the flow. By combining kernel-based flows with optional weighting, KPFF provides a practical and flexible method for high-dimensional nonlinear filtering.

3 Results and Answers

In this section, I'll provide a summary of the answers to the questions posed in the project. I'll include hyperlinks to the relevant code/plot/result on `GitHub` [1] wherever applicable. The `GitHub` repository contains significantly more plots and statistics than what is included in this report. Only the tables and visualizations that are most pertinent to the questions asked are presented here. For additional results and plots, please visit the repository.

Q. I. Linear–Gaussian SSM with Kalman Filter

[1] https://github.com/meamresh/MLCOE_Q2_PF/

a) Implement the Kalman filter for a multidimensional linear–Gaussian state-space model. Synthetic data are generated from a standard LGSSM; see Example 2 in [Doucet(09)].

b) Analyze filtering optimality and numerical stability by comparing filtered means and covariances with the analytical Kalman recursion. Use Joseph-stabilized covariance updates and discuss conditioning effects.

Answer: a. We implemented standard Kalman filter with standard Riccati covariance update and with the Joseph-stabilized form in a linear–Gaussian setting. SSM used in this work is the Linear Gaussian State-Space Model (LGSSM) that provides a canonical framework for discrete-time stochastic processes with linear dynamics and Gaussian noise. The model evolves according to

$$\mathbf{x}_n = \mathbf{A}\mathbf{x}_{n-1} + \mathbf{B}\mathbf{v}_n, \quad \mathbf{v}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_v}), \tag{48}$$

$$\mathbf{y}_n = \mathbf{C}\mathbf{x}_n + \mathbf{D}\mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{n_w}), \tag{49}$$

where $\mathbf{x}_n \in \mathbb{R}^{n_x}$ is the latent state, $\mathbf{y}_n \in \mathbb{R}^{n_y}$ is the observation, and $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are known system matrices of appropriate dimensions. The initial state is Gaussian, $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{m}_0, \mathbf{P}_0)$, and the process and measurement noise are coupled through $\mathbf{B}$ and $\mathbf{D}$, yielding covariances $\mathbf{Q} = \mathbf{B}\mathbf{B}^\top$ and $\mathbf{R} = \mathbf{D}\mathbf{D}^\top$. This parameterization allows exact application of the Kalman filter for recursive state estimation. Synthetic data were generated over 200 time steps with a 4-dimensional state and a 2-dimensional observation model.

Table 1: Comparison of Riccati and Joseph covariance updates.

Metric	Riccati	Joseph
RMSE (filtered)	2.123898	2.123898
RMSE (predicted)	2.728388	2.728388
MAE (filtered)	1.595658	1.595658
Mean NEES	3.761272	3.761272
Mean $\kappa(P_{\text{filt}})$	17.868227	17.868227
Max $\kappa(P_{\text{filt}})$	21.145626	21.145626
Max $\lvert P - P^\top \rvert$	7.1×10^{-7}	4.7×10^{-7}
Min $\lambda_{\min}(P)$	0.527864	0.527864

b. Table 1 summarizes the quantitative performance of the filter using the Riccati covariance update and the Joseph-stablised form. In both cases we get identical filtered and predicted state estimates, with a filtered RMSE of 2.12 and a predicted RMSE of 2.73. All other accuracy parameters are also almost identical for both. This implies that the Joseph stabilization does not disturb the optimal Kalman filtering and preserves the estimation accuracy. The normalized estimation error squared (NEES) was used to evaluate the consistency of the filters, and the obtained values in both cases were 3.76, which is very close to the theoretical expectation of 4, indicating the excellent consistency of covariance estimates.

We assessed the numerical conditioning of the filtered error covariance using the *condition number* $\kappa(P_{\text{filt}})$. We found that for both the Riccati and Joseph updates, the mean condition number was approximately 17.9, with a maximum value of 21.1 over the 200 time steps. These values are relatively small indicating the covariance matrices are very well-behaving, and the inversions made during Kalman gain calculation are stable and reliable. Important thing to note here is that the identical condition numbers observed for the Riccati and Joseph formulations indicate that this particular system itself is not ill-conditioned and that numerical errors do not accumulate significantly over time. In this regime, the standard Riccati is enough and Joseph stabilisation may not be necessary; however, it can provide a safeguard against loss of symmetry and positive semi-definiteness. Visualizations of the statistics in Table 1 are available here, but they were not included in the report because the above arguments and conclusions can be easily drawn from only the table.

Q. II. Nonlinear/Non-Gaussian SSM with EKF/UKF and Particle Filter

a) Design a nonlinear and non-Gaussian SSM, you can use a stochastic volatility model (example 4 in [Doucet(09)]) or nonlinear tracking models(e.g. Range-Bearing observation model)

b) Implement the Extend Kalman filter (EKF) and Unsent Kalman Filter(UKF), discuss linearization accuracy limits and sigma point failures under strong nonlinearity.

c) Implement a standard particle filter for your model. Visualize and discuss issues such as particle degeneracy.

d) Compare PF and EKF/UKF performance. How to evaluate your SSMs? What metrics can you use? In practice, we care about the runtime and memory, could you also compare the runtime and peak memory(CPU/GPU RAM) for each SSM?

Answer: a. Since the Range-Bearing observation model exhibits higher non-linearity and non-Gaussianity compared to the Stochastic Volatility model, primarily due to the presence of square roots and trigonometric functions, We opt for the Range-Bearing model to effectively test the implemented filters in challenging scenarios. For mobile robot localization, we consider a nonlinear state-space model with range and bearing measurements to known landmarks. The state is $\mathbf{x}_n = [x_n, y_n, \theta_n]^\top \in \mathbb{R}^3$, representing position and orientation, and the control input is $\mathbf{u}_n = [v_n, \omega_n]^\top$, the linear and angular velocity. The discrete-time unicycle motion model is

$$\mathbf{x}_{n+1} = \begin{bmatrix} x_n + v_n \Delta t \cos\theta_n \\ y_n + v_n \Delta t \sin\theta_n \\ \theta_n + \omega_n \Delta t \end{bmatrix} + \mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{50}$$

where Δt is the time step and $\mathbf{Q}$ is the process noise covariance.

Observations consist of range and bearing to each of L known landmarks $\ell_i \in \mathbb{R}^2$:

$$\mathbf{z}_n^{(i)} = \begin{bmatrix} \sqrt{(x_n - \ell_{i,x})^2 + (y_n - \ell_{i,y})^2} \\ \arctan 2(y_n - \ell_{i,y},\, x_n - \ell_{i,x}) - \theta_n \end{bmatrix} + \mathbf{v}_n^{(i)}, \quad \mathbf{v}_n^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{51}$$

with $\mathbf{R}$ the measurement noise covariance per landmark. The full measurement vector is obtained by stacking all landmark measurements:

$$\mathbf{z}_n = [\mathbf{z}_n^{(1)\top}, \ldots, \mathbf{z}_n^{(L)\top}]^\top, \quad \mathbf{z}_n \in \mathbb{R}^{2L}.$$

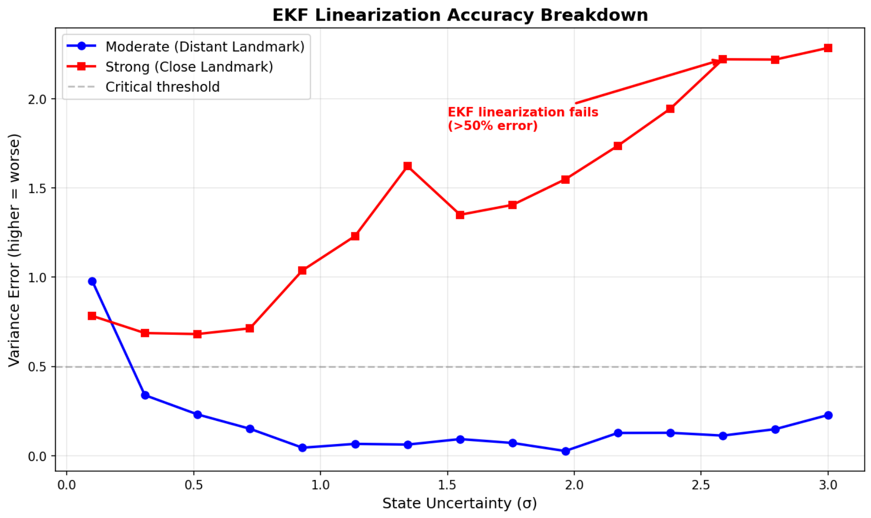


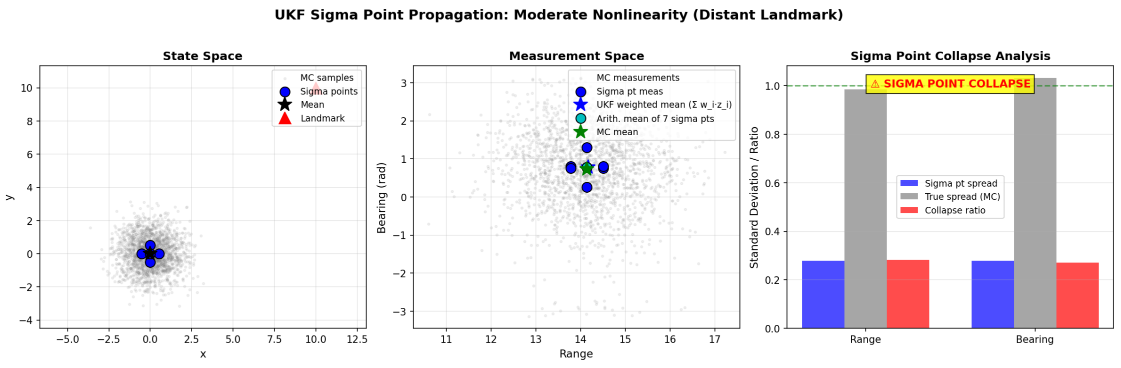
Figure 1: EKF linearization error vs state uncertainty.

This model is nonlinear due to the trigonometric motion model and the range-bearing measurement function, making it ideal for testing nonlinear filtering methods such as the Extended Kalman Filter, Unscented Kalman Filter, or particle and particle flow filters. The Jacobians of the motion and measurement models, $\mathbf{F}_n = \partial f/\partial \mathbf{x}_n$ and $\mathbf{H}_n = \partial h/\partial \mathbf{x}_n$, can be computed analytically and are used in linearization-based filters.

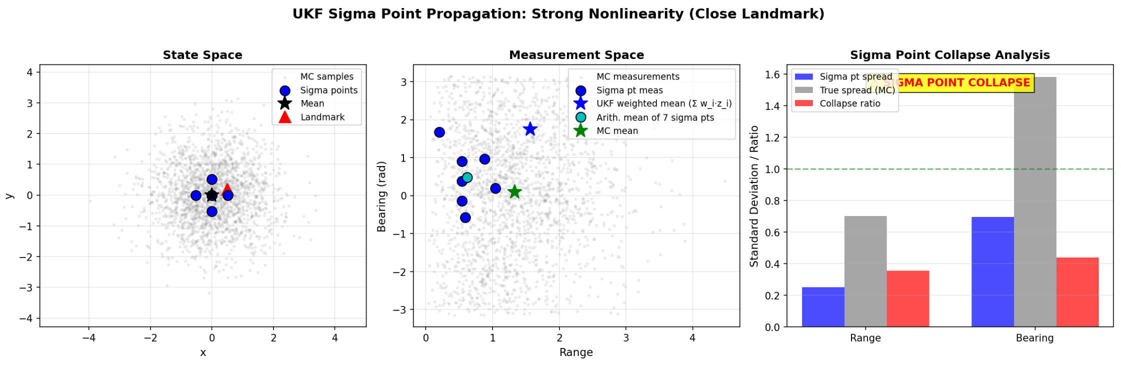
b. We study failure modes of the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) for a 2D range–bearing localization problem. Two geometry-driven nonlinearity regimes are considered: a *moderate* case with a distant landmark and a *strong* case with a close landmark. Performance is evaluated with three diagnostics: (1) EKF linearization variance error, (2) UKF sigma-point collapse ratio (sigma spread / predicted spread), and (3) filter consistency via the Normalized Estimation Error Squared (NEES).

Figure 1 shows the linearization variance error as the uncertainty of the state increases. The red curve is for the strong non-linearity scenario introduced by the arranging landmark to be close. Blue curve represents the case with moderate non-linearity. Concretely, for the distant landmark the EKF variance error across prior spreads $\sigma = \{0.1, 0.5, 1.0, 2.0\}$ was $\{0.937, 0.283, 0.046, 0.077\}$, whereas for the close landmark it was $\{0.867, 0.660, 1.189, 1.669\}$. These large errors near the landmark indicate that the first-order Taylor approximation (Jacobian linearization) no longer captures the true curvature of the measurement map and therefore mispredicts both mean and covariance.

The UKF fails for a different reason: sigma-point collapse. Figure 2 compares sigma-point and Monte Carlo representations in state and measurement space, illustrates the effect of negative weights on the UKF mean, and quantifies sigma-point collapse via measurement-space spread ratios. When the prior is propagated through the nonlinear range–bearing map, the finite set of sigma points can cluster in measurement space and fail to represent the transformed distribution's spread and shape. Notice that the Monte Carlo measurements form a rectangular distribution (bottom middle panel) rather than elliptical. This indicates the range-bearing transformation is so nonlinear that it destroys the Gaussian structure entirely. The UKF's 7 sigma points (clustered in a small region) cannot possibly represent this rect-



(a) Moderate nonlinearity (distant landmark).



(b) Strong nonlinearity (close landmark).

Figure 2: Comparison of UKF failure modes in moderate and high non-linearity regions.

angular spread, leading to severe underestimation of measurement uncertainty. Measured collapse ratios (empirical sigma-spread / predicted spread) were [0.2645, 0.2649] for the distant case (partial collapse) and [0.1440, 0.2195] for the close case (severe collapse). The practical consequence is that the UKF can severely underestimate uncertainty and produce biased means. In our short NEES runs (10 steps) the UKF was overly conservative in the distant case (avg NEES ~ 0.04) but catastrophically inconsistent in the close case (avg NEES $\sim 1.46 \times 10^4$), while the EKF remained overly conservative (avg NEES ~ 0.06 and 0.18, respectively). An exploding NEES for the UKF is a clear sign of representational/numerical failure rather than mere tuning error. Here, it should be emphasized that the UKF is highly sensitive to the model parameters $\{\alpha, \beta, \kappa\}$, and its performance can be significantly improved if these parameters are chosen judiciously. To address this, we have created an experiment file that automatically tunes (when executed with the `-tuning` flag) these parameters and identifies the optimal values.

c. We evaluated particle degeneracy in a nonlinear range–bearing tracking particle filter with N=1000 particles over T=150 time steps, using ESS-triggered resampling with threshold 0.5N. Degeneracy was monitored using Effective Sample Size (ESS), normalized weight entropy, the weight distribution at the most degenerate time, and the particle spread in (x,y) state space. ESS is also sometimes represented by N_{eff}, defined as $N_{eff} = 1/\sum_i \bar{w}_i^2$.

The experimental results, as shown in Figure 3, reveal that the particle filter encountered

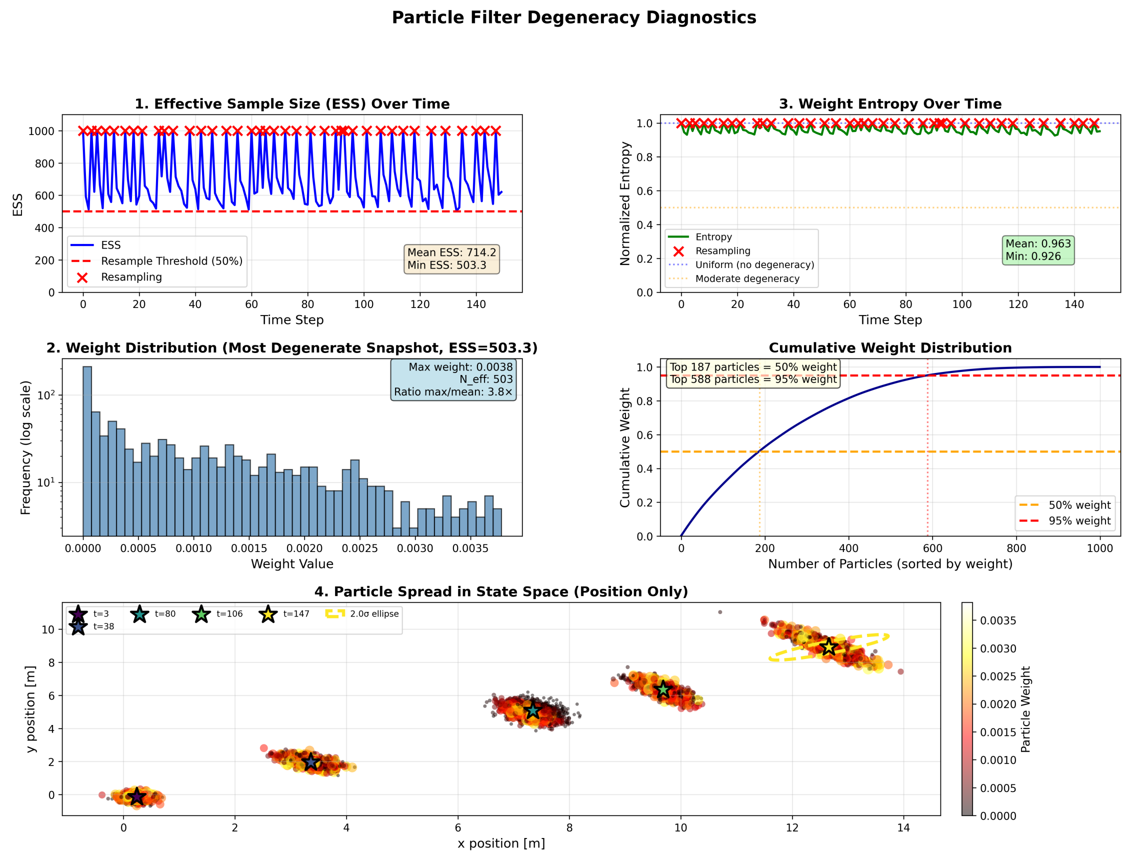


Figure 3: Particle degeneracy analytics of Particle Filter.

degeneracy at a relatively high frequency, necessitating 39 resampling interventions over the 150-step simulation—corresponding to a 26.0% resampling rate. Each intervention was triggered when the Effective Sample Size (ESS) fell below the 50% threshold, indicating a substantial increase in particle weight variance and a reduced effective representation of the posterior distribution. This recurrent behavior is clearly visible in the characteristic “sawtooth” ESS profile, where the filter progressively approaches degeneracy before being reset by resampling.

While the filter experienced degeneracy on multiple occasions, diagnostic metrics indicate that these events were effectively managed using the adopted resampling strategy and did not result in filter failure. The mean normalized weight entropy of 0.9630 and the minimum ESS of 503.30 suggest that resampling was performed before severe weight collapse occurred. Consistent with this observation, the most degenerate snapshot exhibits a maximum particle weight of only 0.0038, indicating that no single particle dominated the posterior. Spatially, the particle distribution consistently tracks the true trajectory, with the 2σ covariance ellipses capturing the state uncertainty throughout the curved maneuvers, supporting the conclusion that the filter maintained a representative approximation of the posterior over the full 150-step trajectory.

d. The experimental evaluation of EKF, UKF, and PF using the Range-Bearing SSM reveals a clear hierarchy in estimation performance. The filters were evaluated using a multi-

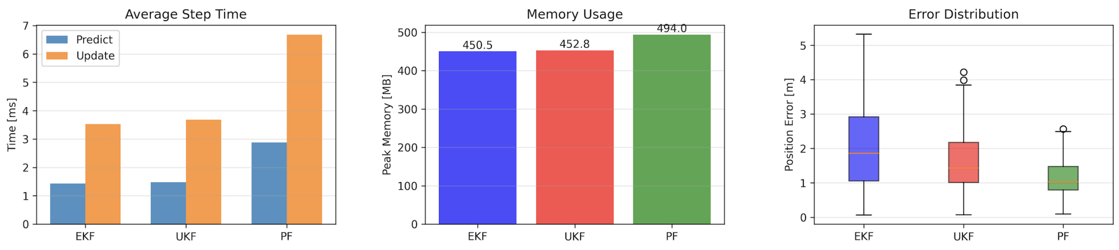


Figure 4: Comparison between EKF, UKF, and PF's performance and memory usage.

dimensional metric suite categorized into accuracy, consistency, and efficiency. Root Mean Square Error (RMSE) was used to quantify the average deviation from the true trajectory. To assess the "health" of the filters, the Normalized Estimation Error Squared (NEES) and Normalized Innovation Squared (NIS) were employed; these metrics determine if the filter's internal uncertainty (covariance) accurately reflects the actual error. A high NEES, as seen in the EKF results, signals a "dishonest" filter that underestimates its error, whereas a NEES near the state dimension indicates a well-tuned, consistent estimator.

The results are summarised in Table 2 and visualised in Figure 4. The PF achieved the highest accuracy with a Position RMSE of 0.9060m, representing a 45.4% improvement over the baseline EKF. Beyond pure accuracy, the PF demonstrated superior statistical consistency, maintaining a Mean NEES of 1.25—the closest to the theoretical ideal for a 3D state—and a 100% consistency rate. In contrast, the EKF showed significant overconfidence with a Mean NEES of 40.03, indicating that its linear approximations failed to capture the true uncertainty of the nonlinear maneuvers. The UKF served as a robust middle ground, improving accuracy by 20.7% over the EKF and achieving a 76.0% consistency rate by utilizing sigma points to better represent the state distribution.

Table 2: Filter Performance Comparison

Filter	Pos RMSE [m]	Head RMSE	Mean NEES	Mean NIS	Time [s]	RAM [MB]
EKF	1.6605	0.3972	40.03	5.11	0.990	450.5
UKF	1.3161	0.3311	6.40	4.73	1.032	452.8
PF	0.9060	0.2878	1.25	9.77	1.912	494.0

In practical terms, such as computational cost and runtime, the results reveal a substantial trade-off between computational overhead and tracking quality. The EKF remains the most efficient, requiring only 0.990s total runtime and minimal memory. The UKF adds negligible overhead, running only 4% slower than the EKF, making it an ideal candidate for systems with limited compute but a need for higher accuracy. However, the Particle Filter is the most demanding, with a total runtime of 1.912s (1.93x slower than EKF) and a peak memory usage of 494.0 MB. The increased latency for the PF (compared to the EKF) suggests that while it provides the most "honest" and accurate estimate, its use is best suited for high-performance hardware where the cost of a 10% increase in RAM and doubled processing time is justifiable.

Additionally, a scaling analysis was conducted to investigate how the PF behaves with varying numbers of particles .It demonstrated that the PF's statistical consistency improved drastically from 50.0% to 99.3% as the particle count increased from 500 to 10,000. Over this range, the Position RMSE decreased by 59% (from 2.25m to 0.93m) and the Mean NEES converged from an overconfident 180.23 down to a consistent 1.77. While increasing the count further to 20,000 provided a marginal gain in consistency (99.5%), it resulted in a 112% increase in total runtime compared to the 500-particle baseline (1.86s vs 0.88s), marking 10,000 particles as the optimal balance between accuracy and computational cost.

Q. 2. Deterministic and Kernel Flows

a) Study the Exact Daum-Huang (EDH) flow and Local Exact Daum-Huang (LEDH) flow, (see [Daum(10)] and [Daum(11)]), and the invertible particle flow particle filter (PF-PF) framework (see Li(17)). Replicates the main results in Li(17).

b) Implement the kernel-embedded particle flow filter in an RKHS following Hu(21). Then compare the scalar kernel and diagonal matrix-valued kernel. Use experiments to demonstrate the matrix-valued kernel can prevent collapse of observed-variable marginals in high-dimensional, plot similar figures as figure 2-3 in Hu(21).

c) Compare EDH,LEDH, and kernel PFF on the SSM you designed in last particle filter question. Analyze when each method excels or fails(nonlinearity, observation sparsity, dimension, conditioning). Include stability diagnostics(flow magnitude, Jacobian conditioning).

Answer: a. In Li(17) [10], the relevant filters that are relevant to this project are EDH, LEDH, PFPF LEDH, and PFPF EDH, along with the EKF, UKF, and PF. These filters are tested on the multi-target acoustic tracking SSM. For the multi-target acoustic tracking scenario considered in this work, the state at time k consists of the concatenated position and velocity vectors of C targets,

$$\mathbf{x}_k = [x_{1,k}, y_{1,k}, v_{x1,k}, v_{y1,k}, \dots, x_{C,k}, y_{C,k}, v_{xC,k}, v_{yC,k}]^\top \in \mathbb{R}^{4C}, \tag{52}$$

evolving under a constant velocity motion model:

$$\mathbf{x}_{k+1} = \mathbf{F}\,\mathbf{x}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \tag{53}$$

where $\mathbf{F}$ is block-diagonal with 4×4 constant-velocity subblocks and $\mathbf{Q}$ is the block-diagonal process noise covariance. The measurement model is nonlinear: each of the N_s fixed sensors receives a scalar acoustic amplitude that depends on the inverse squared distances to all targets,

$$z_{n,k} = \sum_{i=1}^{C} \frac{\psi}{\|\mathbf{p}_{i,k} - \mathbf{s}_n\|^2 + d_0} + v_{n,k}, \quad v_{n,k} \sim \mathcal{N}(0, \sigma_w^2), \tag{54}$$

where $\mathbf{p}_{i,k} = [x_{i,k}, y_{i,k}]^\top$, $\mathbf{s}_n$ is the n-th sensor position, ψ is an amplitude scale factor, d_0 prevents singularities at zero distance, and σ_w^2 is measurement noise variance. The resulting posterior $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ is high-dimensional and non-Gaussian, presenting challenges for standard particle filters due to weight degeneracy; this motivates the use of particle flow and

Table 3: RMSE, ESS, and Runtime Comparison.

Algorithm	RMSE	Particle num.	Avg. ESS	Time [s]
PFPF (LEDH)	2.4915 ± 1.4225	500	8.09	10.8571 ± 0.3172
PFPF (EDH)	5.1450 ± 3.4483	500	39.14	2.4559 ± 0.0348
LEDH	3.5432 ± 0.9524	500	N/A	5.1965 ± 0.2601
EDH	2.9811 ± 2.2232	500	N/A	2.2064 ± 0.0456
PF	3.3890 ± 1.9974	5000	5.59	0.1533 ± 0.0158
EKF	13.8402 ± 16.2770	N/A	N/A	0.1052 ± 0.0020
UKF	11.7729 ± 15.1089	N/A	N/A	0.1196 ± 0.0060

invertible transport methods that construct efficient proposal distributions by migrating particles toward regions of high posterior density.

The most important finding of Li(17) is summarised in their Table I, because it evaluates the proposed invertible particle-flow particle filter (PF-PF) in the kind of setting the method is designed for: a nonlinear, multi-target tracking problem where standard filters struggle and brute-force particle filters require enormous numbers of samples. In this realistic scenario, PF-PF with LEDH achieves the lowest tracking error among practical methods, maintains a higher effective sample size than a bootstrap PF with far fewer particles. By contrast, later tables include linear-Gaussian experiments where Kalman-type filters are theoretically optimal, so outperforming them is neither expected nor central to the paper's claim.

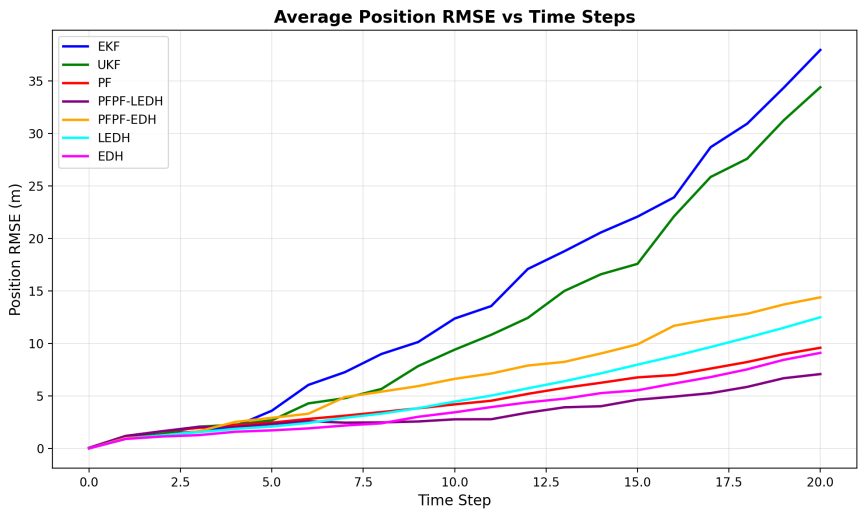


Figure 5: A comparison of average position RMSE values for different filters, as a function of time steps.

Therefore, a comparative study was conducted for all the relevant filters using the SSM parameters outlined in Li (17). The primary results are reproduced in Table 3. Moreover, Figure 5 shows the mean position RMSE (in meters) computed over all Monte Carlo runs for each filter at every time step. Results are reported for EKF, UKF, PF, PF-PF (LEDH/EDH), and standalone LEDH/EDH filters. We can clearly see that PFPF (LEDH) outperforms every other filter in this scenario with just 500 particles confirming the main findings of Li(17).

b. To evaluate the effectiveness of the Kernel-Embedded Particle Flow Filter, we implemented the filter within a Reproducing Kernel Hilbert Space (RKHS) following the methodology of Hu and van Leeuwen (2021). They consider the Lorenz–96 system as a nonlinear, high-dimensional state-space model commonly used in data assimilation experiments. The state at time n is $\mathbf{x}_n \in \mathbb{R}^d$, where d denotes the system dimension and $F > 0$ is a constant forcing parameter. The continuous-time dynamics are governed by

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad i = 1, \dots, d,$$

with cyclic boundary conditions.

The system is discretized using a fourth-order Runge–Kutta scheme with time step Δt, resulting in the discrete-time transition model

$$\mathbf{x}_{n+1} = f(\mathbf{x}_n) + \mathbf{w}_n, \quad \mathbf{w}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}),$$

where $f(\cdot)$ denotes the RK4 propagation operator. This model is strongly nonlinear and chaotic, and is widely used as a benchmark for nonlinear filtering and data assimilation methods.

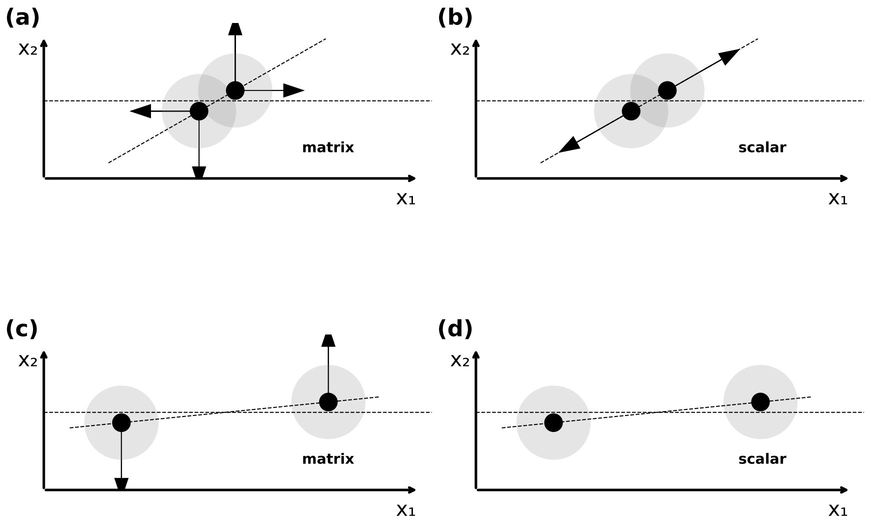


Figure 6: Comparison of the divergence of kernel between (a, c) the matrix-valued kernel and (b, d) the scalar kernel.

Figure 6 is replication of the schematic results (analogous to Figure 2 in the original paper) demonstrating the fundamental difference in how these kernels handle particle interactions. While a scalar kernel applies a uniform repulsion force based on the total Euclidean distance between particles, the diagonal matrix-valued kernel allows for dimension-specific repulsion. This flexibility ensures that particles are effectively pushed apart in observed dimensions even when their distances in unobserved dimensions might otherwise “mask" the need for repulsion, thereby maintaining a more representative ensemble spread during the flow.

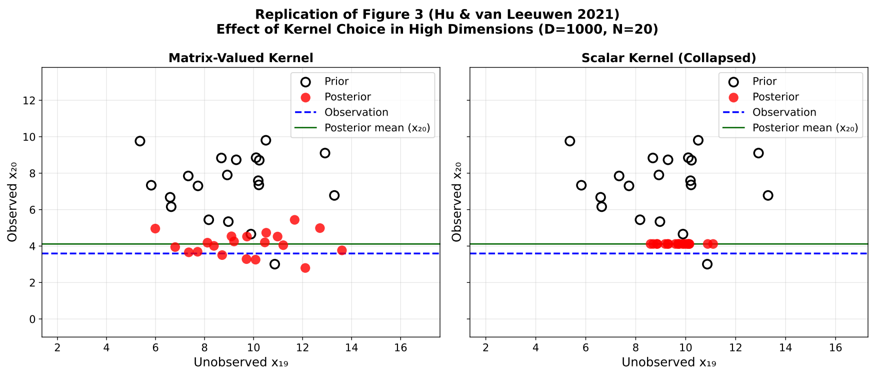


Figure 7: The prior and posterior marginal distribution of the variable x_{19} (unobserved component) and x_{20} (observed component) before and after data assimilation, using matrix-valued kernel and scalar kernel.

The most striking result of this analysis is the matrix kernel's ability to prevent the collapse of observed-variable marginals. In the high-dimensional Lorenz-96 experiment, as seen in Figure 7, the scalar kernel caused the ensemble to collapse almost entirely in the observed dimension (x_{20}), resulting in a diversity ratio of 0.000 (spread of 0.001). In contrast, the matrix-valued kernel preserved significant ensemble diversity with a diversity ratio of 0.371 (spread of 0.642). This preservation of variance allows the filter to remain "active" and receptive to new information, whereas the scalar kernel's collapse leads to a rigid, uninformative ensemble.

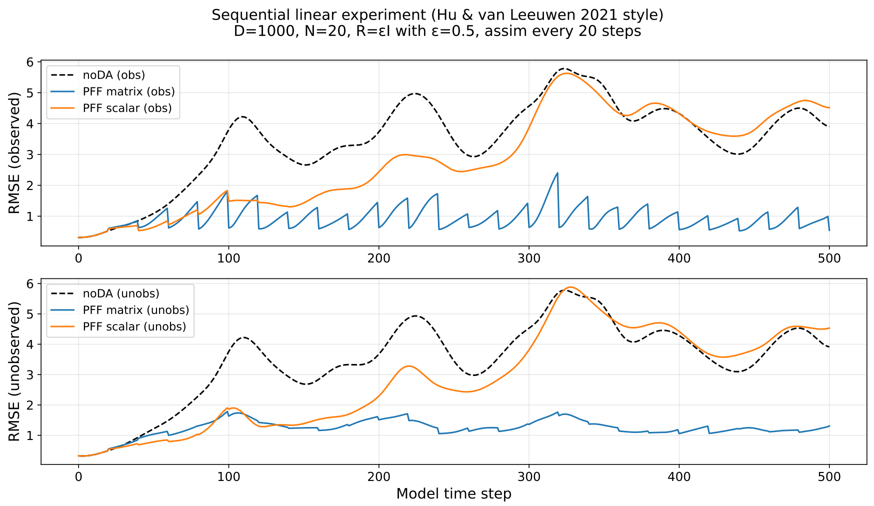


Figure 8: Sequential RMSE over 500 steps comparing matrix-valued and scalar kernels, showing stable low error for the matrix-valued approach and divergence of the scalar kernel.) before and after data assimilation, using matrix-valued kernel and scalar kernel.

In Figure 8, the sequential RMSE time series further confirms the superiority of the matrix-valued approach. Over 500 model time steps, the matrix-valued kernel maintained a stable and low RMSE of 1.167, representing a massive improvement over the no-assimilation (noDA) baseline of 3.913. Conversely, the scalar kernel suffered from filter divergence; its

RMSE steadily climbed from 1.772 at $t = 100$ to 4.528 at $t = 500$. By the end of the simulation, the scalar kernel's performance was actually worse than the noDA baseline, illustrating that the marginal collapse observed in the snapshots eventually leads to a total failure of the state estimation process in high-dimensional settings.

c. We assess filter performance across different challenges, such as nonlinearity, sparse observations, dimensionality, and conditioning using the multi-target acoustic tracking example from the previous question. Each target has 4D state (e.g. position and velocity); measurements are received from a sensor grid. Scenarios vary the number of targets (1–8), observation rate (30–100%), sensor count and layout, process/observation noise, and correlation. Each scenario is run multiple times (e.g. several Monte Carlo runs) with fixed filters and hyperparameters.

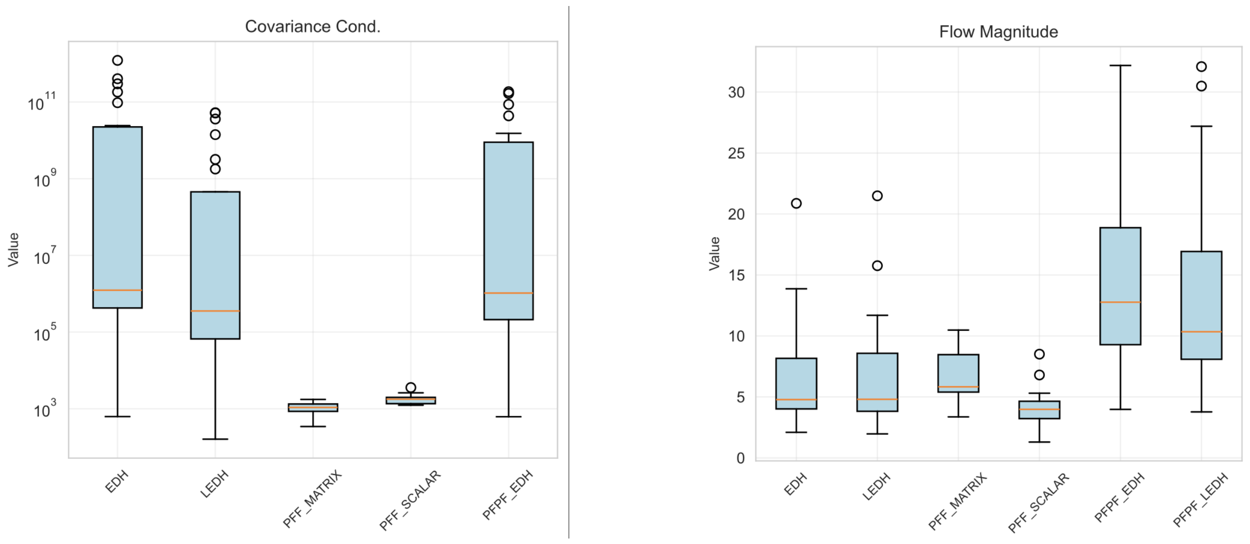


Figure 9: Box plots showing covariance stability and flow magnitude for each filters across all scenarios/cases.

We assess numerical conditioning of each filter's covariance matrix P via its condition number $\kappa(P)$. For each run we record the mean condition number over time, then for each filter we collect these scenario-level means and plot their distribution in a box plot (one box per filter, over all scenarios), see Figure 9. The y-axis is logarithmic. Values above roughly 10^7 indicate ill-conditioning and risk of numerical instability or filter divergence; the plot shows which filters tend to have well- or ill-conditioned covariances across the scenario set. Similary, the flow magnitude box plot shows the distribution of the time-averaged particle displacement per update for each filter across scenarios, to compare the strength of the measurement update and related stability.

We also plot a failure diagnosis heatmap. Figure 10 two panels: one shows RMSE and success/failure by scenario and filter (with log-scaled color), and the other shows the number of stability issues (0–5) per scenario–filter pair, so that performance failures can be compared with stability diagnostics in one view.

As plotted in Figure 9, the EDH and LEDH variants exhibit relatively high particle flow, around 15–20, indicating that particles move aggressively at each update. In contrast, the PFF kernel variants show lower flow, roughly 5–10, corresponding to gentler and more stable particle updates. Physically, a high flow value suggests that the filter is "surprised" by incoming measurements, requiring frequent and significant repositioning of particles to match the observed data. The same trend is evident in the covariance condition plot, where PFF kernel variants exhibit smaller condition numbers compared to EDH and LEDH. Figure 10 further illustrates the stability of PFF kernels, as indicated by their consistent performance in terms of lower RMSE (left panel) and minimal stability issues (right panel).

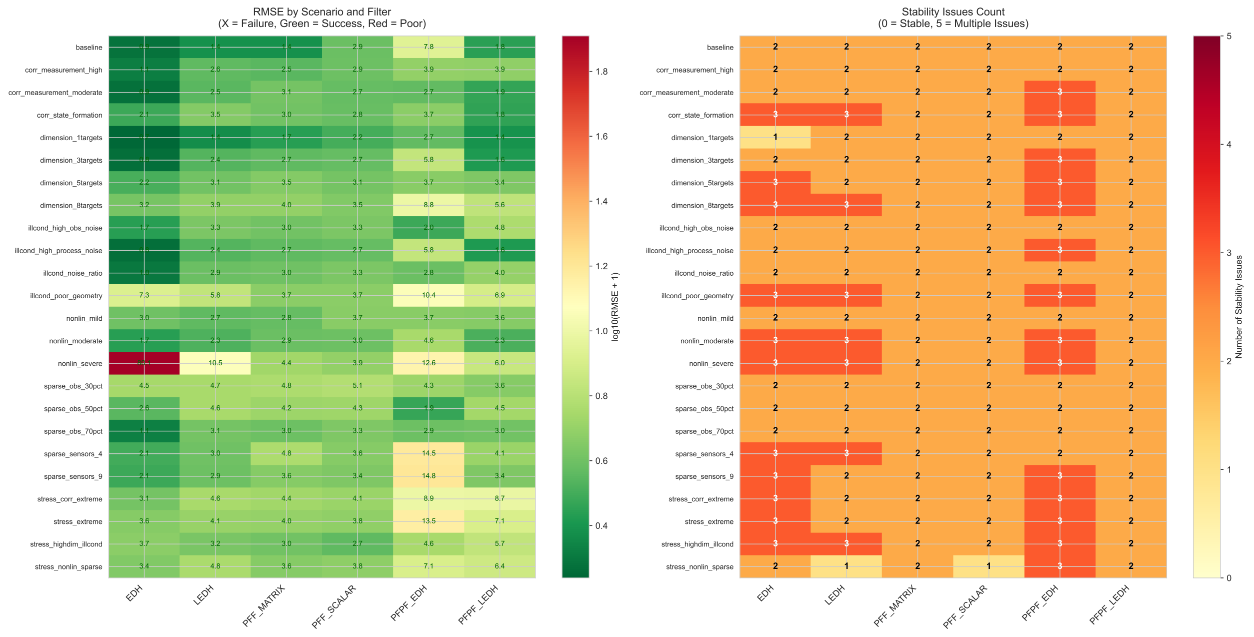


Figure 10: Left panel: RMSE by scenario (rows) and filter (columns). Color is $\log_{10}(\text{RMSE}+1)$ (red = worse, green = better). Right panel: Number of stability issues (0–5) by scenario and filter. Each count is how many of five diagnostic conditions were exceeded (ill-conditioned Jacobians/covariance, low ESS, high flow magnitude).

4 Conclusion

This report investigated state estimation across progressively more challenging regimes, beginning with linear–Gaussian models where the Kalman filter is optimal, extending to nonlinear and non-Gaussian tracking scenarios that expose the limitations of Gaussian approximations, and culminating in modern deterministic transport and kernel-based particle flow methods designed for high-dimensional inference. In the linear setting, Riccati and Joseph covariance updates were shown to yield identical accuracy and consistency, with Joseph stabilization acting purely as a numerical safeguard rather than altering optimality. For nonlinear range–bearing localization, systematic failure modes of the EKF and UKF were revealed through linearization error, sigma-point collapse, and NEES diagnostics, while the particle filter delivered the most statistically consistent and accurate estimates at the cost of increased computation and memory. Scaling studies further highlighted the classical accuracy–efficiency trade-off inherent to Monte-Carlo methods. Building on these foundations, deterministic particle-flow approaches were examined through replication of Li (2017) and Hu (2021), demonstrating that invertible flows such as PF-PF can dramatically outperform conventional filters in nonlinear multi-target tracking, and that matrix-valued kernels in RKHS flows prevent marginal collapse and stabilize filtering in high dimensions. Comparative stress-testing across nonlinearity, sparsity, dimensionality, and conditioning confirmed that while EDH/LEDH flows can become numerically aggressive, kernel-based particle flow filters provide smoother updates, better covariance conditioning, and superior robustness. Taken together, the experiments illustrate a coherent progression: as models depart from linear-Gaussian assumptions and dimensionality grows, success increasingly depends on transport-based proposals that reshape particle ensembles deterministically rather than relying on resampling alone, positioning particle-flow and kernel-embedded methods as powerful tools for next-generation nonlinear filtering and data assimilation.

References

[1] Mike West and Jeff Harrison. *Bayesian Forecasting and Dynamic Models.* Springer Series in Statistics. Springer-Verlag, New York, 2 edition, 1997.

[2] Genshiro Kitagawa and Winfried Gersch. *Smoothness Priors Analysis of Time Series.* Springer-Verlag, New York, 1996.

[3] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[4] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, 1979.

[5] Simon J. Julier. A skewed approach to filtering. In *Proceedings of SPIE: Signal and Data Processing of Small Targets*, volume 3373 of *AeroSense '98*, pages 54–65, Orlando, FL, USA, 1998.

[6] Arnaud Doucet, Nando de Freitas, and Neil J. Gordon. An introduction to sequential monte carlo methods. In Arnaud Doucet, Nando de Freitas, and Neil J. Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 3–14. Springer, New York, NY, USA, 2001.

[7] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[8] Fred Daum, Jim Huang, and Arjang Noushin. Exact particle flow for nonlinear filters. In *Proceedings of SPIE: Signal Processing, Sensor Fusion, and Target Recognition XIX*, volume 7697, pages 92–110, Orlando, FL, USA, 2010. SPIE.

[9] T. Ding and Mark J. Coates. Implementation of the daum–huang exact flow particle filter. In *Proceedings of the IEEE Statistical Signal Processing Workshop (SSP)*, pages 257–260, Ann Arbor, MI, USA, August 2012.

[10] Yunpeng Li and Mark Coates. Particle filtering with invertible particle flow. *IEEE Transactions on Signal Processing*, 65(15):4102–4116, 2017.

[11] Chih-Chi Hu and Peter Jan van Leeuwen. A particle flow filter for high-dimensional system applications. *Quarterly Journal of the Royal Meteorological Society*, 147(737):2352–2374, 2021.