

An Analysis and Experimental Evaluation of the Variational Particle Flow Particle Filter

Prepared for ECE 199: Independent Study for Undergraduates

Anthony Nguyen
University of California, San Diego
San Diego, CA 92093
ann048@ucsd.edu

June 13, 2025

1 Introduction

This work aims to study and analyze the numerical integration methods implementing the variational formulation of the particle flow particle filter (VPFPF), introduced in [7], as a way to solve the Bayesian estimation problem of finding the posterior distribution $p(\mathbf{x}|\mathbf{z})$ from a prior density function $p(\mathbf{x})$, given an observation \mathbf{z} and a likelihood density function $p(\mathbf{z}|\mathbf{x})$.

Bayesian estimation provides a powerful way to estimate unknown quantities while explicitly accounting for uncertainty through estimating the effects of new information \mathbf{z} on probability distributions $p(\mathbf{x}|\mathbf{z})$, rather than deterministic values. Due to the often complicated nature of the associated probability distributions in Bayesian estimation, finding the solution to the problem, the true posterior $p(\mathbf{x}|\mathbf{z})$, is often impossible. As a result, many methods, like particle flow particle filtering (PFPF), variational inference, and VPFPF, which connects the two methods, have arisen to provide ways to approximate this true posterior.

The PFPF method [3] estimates the posterior distribution by propagating points $\{\mathbf{x}_i\}_{i=1}^n$ sampled from the prior $p(\mathbf{x})$ according to differential equations satisfying the Fokker-Planck equation. This process provides a discrete representation of the posterior that grows in accuracy with more samples from the prior. The propagated samples then function as samples from the true posterior $p(\mathbf{x}|\mathbf{z})$. While the movement of particles in PFPF avoids the particle degeneracy, the existence of negligible particle weights, commonly seen in sequential Monte-Carlo methods (traditional particle filters), by moving the particles instead of changing their weights, PFPF particles can suffer from poor positioning and diversity due to the numerical approximations made to implement the flow, decreasing the accuracy of the approximation.

Variational inference (VI) [4] introduces a chosen tractable proxy distribution $q(\mathbf{x})$ to be optimized to closely fit the true posterior $p(\mathbf{x}|\mathbf{z})$ by minimizing the Kullback-Leibler (KL) divergence between the two distributions. This resulting optimization problem can often be written entirely in terms of known and tractable distributions, like $p(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x})$, avoiding computation of intractable distributions like $p(\mathbf{z})$. VI's formulation of inference as an optimization problem and production of an expression for the approximate posterior make it popular in fields like deep generative learning. However, KL divergence's tendency to ignore low values of $q(\mathbf{x})$ result in mode-seeking behavior, the tendency to collapse onto high probability regions, during optimization, making VI a poor choice for approximating multi-modal distributions.

The VPFPF method [7] connects PFPF with variational inference, introducing methods to find the variational posterior distribution $q(\mathbf{x})$ by propagating the parameters of the prior $p(\mathbf{x})$ along differential equations. This effectively solves PFPF’s particle diversity issues, provides an expression for the approximated posterior, and can simplify the finding of approximations for multi-modal distributions due to the formulation of the variational inference optimization problem as a numerical integration problem, which is easier to solve when using a mixture distribution as the variational density $q(\mathbf{x})$.

The main contribution of this work is the analysis and comparison of many different implementation methods for VPFPF, namely methods using analytically-computed and autodifferentiated derivatives, along with derivative-free reformulations of the VPFPF parameter flows using Stein’s Lemma [5], introduced in [7]. To help streamline the implementation of the aforementioned methods, this work introduces formulations modified continuous and discrete VPFPF parameter flows and discusses key implementation insights, like the need to remove affine terms from the test function $f(\mathbf{x})$ when approximating the expectation in Stein’s Lemma for increased numerical stability.

This paper is organized with the following structure: Section 2 provides necessary background, including a more detailed formulation of the Bayesian estimation problem that underpins the discussed solution methods and a review of VPFPF. Section 3 unveils an alternative formulation of the parameter flows for simpler differentiation, along with a discrete formulation of the parameter flows. Section 4 discusses important approximations, assumptions, and methods used to implement the VPFPF parameter flows, including the importance of removing constant terms from your chosen function when using approximating Stein’s Lemma. Finally, Section 5 formulates a specific Nonlinear Observation Model Bayesian Estimation problem and discusses and compares analytical, to the problem.

2 Background

This paper considers a general Bayesian estimation problem where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$ are random variables. The probability density function (PDF) of \mathbf{x} , $p(\mathbf{x})$, is taken to be the prior, and the conditional distribution of \mathbf{z} given \mathbf{x} , $p(\mathbf{z}|\mathbf{x})$, is the likelihood. Then, according to Bayes’ Theorem [2], the posterior distribution is given by

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}. \quad (1)$$

In the context of this work, \mathbf{z} would represent an observation or measurement, and \mathbf{x} would represent the states used to generate observations \mathbf{z} .

2.1 Parameter Flows of the Variational Formulation of the Particle Flow Particle Filter (VPFPF)

As discussed in the introduction (Section 1), VPFPF is a very effective method to solve the Bayesian estimation problem, providing an approximate expression (the variational density) $q(\mathbf{x})$ for the posterior $p(\mathbf{x}|\mathbf{z})$ and solving for $q(\mathbf{x})$ using numerical integration.

For the purposes of this work, the variational density $q(\mathbf{x})$ will be Gaussian and will be notated as $q(\mathbf{x}; \boldsymbol{\alpha}_t)$, where $\boldsymbol{\alpha}_t$ from now on. This assumption is also made in [7], giving rise to the Gaussian Fisher-Rao particle flows, which are formulated below, given that $V(\mathbf{x}; \boldsymbol{\alpha}_t) = \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) + \phi(\mathbf{x})$, with $q(\mathbf{x}; \boldsymbol{\alpha}_t) = p_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ and $\phi(\mathbf{x}) = -\ln(p(\mathbf{x}, \mathbf{z}))$:

$$\frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 V(\mathbf{x}; \boldsymbol{\alpha}_t)}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right], \quad \frac{d\boldsymbol{\mu}_t}{dt} = -\boldsymbol{\Sigma}_t \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial V(\mathbf{x}; \boldsymbol{\alpha}_t)}{\partial \mathbf{x}^\top} \right], \quad (2)$$

3 Alternative formulations of the VPFPF parameter flows

When implementing the original Gaussian Fisher-Rao parameter flows in Equation (2), one of the central challenges is the computation, especially analytically, of the derivatives within the expectations. This section focuses on an alternative formulation of the VPFPF parameter flows that simplifies differentiation and a Stein’s Lemma [5] formulation, introduced in [7], that entirely eliminates derivative computation. A discrete formulation of the parameter flows will also be introduced, giving the option for a loop-based implementation and the simplicity and tunability that comes with it.

3.1 VPFPF parameter flow formulation for simpler differentiation

The parameter flows in Equation (2) can be rewritten with derivatives of $\phi(\mathbf{x}) = -\ln(p(\mathbf{x}, \mathbf{z}))$ instead of $V(\mathbf{x}; \boldsymbol{\alpha}_t) = \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) + \phi(\mathbf{x})$, which simplifies the computation process:

$$\frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] - \boldsymbol{\Sigma}_t^{-1}, \quad \frac{d\boldsymbol{\mu}_t}{dt} = -\boldsymbol{\Sigma}_t \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right] \quad (3)$$

The derivation of these parameter flows from those in Equation (2) is shown in the proof below.

Proof With $V(\mathbf{x}; \boldsymbol{\alpha}_t) = \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) + \phi(\mathbf{x})$, $q(\mathbf{x}; \boldsymbol{\alpha}_t) = p_{\mathcal{N}}(\mathbf{x}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$, and $\phi(\mathbf{x}) = -\ln(p(\mathbf{x}, \mathbf{z}))$, the gradient and hessian expressions and Equation (3) can be rewritten by evaluating the derivatives of $\ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))$, which is defined with the following fixed form in VPFPF:

$$\begin{aligned} \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) &= \ln \left(\frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_t|}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1}(\mathbf{x} - \boldsymbol{\mu}_t)} \right) \\ &= -\ln \left(\frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_t|}} \right) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}_t^{-1}(\mathbf{x} - \boldsymbol{\mu}_t) \end{aligned}$$

The gradient and hessian of $\ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))$ are then given by the following:

$$\frac{\partial \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top} = -\boldsymbol{\Sigma}_t^{-1}(\mathbf{x} - \boldsymbol{\mu}_t), \quad \frac{\partial^2 \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top \partial \mathbf{x}} = -\boldsymbol{\Sigma}_t^{-1}$$

Taking the expectation of the above gradient and hessian expressions with respect to $q(\mathbf{x}; \boldsymbol{\alpha}_t)$ results in

$$\begin{aligned} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top} \right] &= \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} [-\boldsymbol{\Sigma}_t^{-1}(\mathbf{x} - \boldsymbol{\mu}_t)] = -\boldsymbol{\Sigma}_t^{-1} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} [\mathbf{x}] + \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\mu}_t \\ &= -\boldsymbol{\Sigma}_t^{-1} \boldsymbol{\mu}_t + \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\mu}_t = 0 \\ \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] &= \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} [-\boldsymbol{\Sigma}_t^{-1}] = -\boldsymbol{\Sigma}_t^{-1}. \end{aligned}$$

Now, the parameter flows from Equation (2) can be rewritten with the evaluated expectations above in mind:

$$\begin{aligned} \frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} &= \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 V(\mathbf{x}; \boldsymbol{\alpha}_t)}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] = \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 [\ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) + \phi(\mathbf{x})]}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] \\ &= \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] + \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] = \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] - \boldsymbol{\Sigma}_t^{-1} \end{aligned}$$

$$\begin{aligned}
\frac{d\boldsymbol{\mu}_t}{dt} &= -\boldsymbol{\Sigma}_t \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial V(\mathbf{x}; \boldsymbol{\alpha}_t)}{\partial \mathbf{x}^\top} \right] = -\boldsymbol{\Sigma}_t \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial [\ln(q(\mathbf{x}; \boldsymbol{\alpha}_t)) + \phi(\mathbf{x})]}{\partial \mathbf{x}^\top} \right] \\
&= -\boldsymbol{\Sigma}_t \left(\mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \ln(q(\mathbf{x}; \boldsymbol{\alpha}_t))}{\partial \mathbf{x}^\top} \right] + \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right] \right) \\
&= -\boldsymbol{\Sigma}_t \left(0 + \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right] \right) = -\boldsymbol{\Sigma}_t \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right]
\end{aligned}$$

3.2 Derivative-free (Stein's Lemma) parameter flow formulation

The computation of derivatives in the VPFPPF parameter flows in Equation (2) can also be entirely avoided by rewriting them using Stein's Lemma [5], which, in this paper's notation, is given by

$$\mathbb{E}_q \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right] = \boldsymbol{\Sigma}^{-1} \mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu}) f(\mathbf{x})], \quad (4)$$

and applying Stein's lemma twice results in

$$\mathbb{E}_q \left[\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] = \boldsymbol{\Sigma}^{-1} \mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top f(\mathbf{x})] \boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1} \mathbb{E}_q[f(\mathbf{x})], \quad (5)$$

Substituting the above expressions into the parameter flows in Equation (2) gives rise to the following:

$$\frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \boldsymbol{\Sigma}_t^{-1} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)}[(\mathbf{x} - \boldsymbol{\mu}_t)(\mathbf{x} - \boldsymbol{\mu}_t)^\top V(\mathbf{x}; \boldsymbol{\alpha}_t)] \boldsymbol{\Sigma}_t^{-1} - \boldsymbol{\Sigma}_t^{-1} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)}[V(\mathbf{x}; \boldsymbol{\alpha}_t)], \quad (6)$$

$$\frac{d\boldsymbol{\mu}_t}{dt} = -\mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)}[(\mathbf{x} - \boldsymbol{\mu}_t) V(\mathbf{x}; \boldsymbol{\alpha}_t)]. \quad (7)$$

3.3 Discrete formulation of the VPFPPF flows

Additionally, the flow equations in Equation (3) can be written as the following discrete update, similar to the updates seen in [1], where λ_t and γ_t are learning rates at the t th discrete iteration:

$$\boldsymbol{\Sigma}_{t+1}^{-1} = \boldsymbol{\Sigma}_t^{-1} + \lambda_t \left(\mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] - \boldsymbol{\Sigma}_t^{-1} \right), \quad \boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t - \gamma_t \boldsymbol{\Sigma}_{t+1} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right]. \quad (8)$$

These discrete updates can be written as the continuous updates in Equation (2) if the learning rates $\lambda_t, \gamma_t \rightarrow 0$, and this is shown in the proof below:

Proof Assuming the learning rates $\lambda_t, \gamma_t \rightarrow 0$, the updates in Equation (8) can be written as ordinary differential equations (ODEs):

$$\lim_{\lambda_t \rightarrow 0} \frac{\boldsymbol{\Sigma}_{t+1}^{-1} - \boldsymbol{\Sigma}_t^{-1}}{\lambda_t} = \frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] - \boldsymbol{\Sigma}_t^{-1} \quad (9)$$

$$\lim_{\gamma_t \rightarrow 0} \frac{\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_t}{\gamma_t} = \frac{d\boldsymbol{\mu}_t}{dt} = -\boldsymbol{\Sigma}_{t+1} \mathbb{E}_{q(\mathbf{x}; \boldsymbol{\alpha}_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right] \quad (10)$$

Now, since $\lambda_t \rightarrow 0$, $\Sigma_{t+1}^{-1} = \Sigma_t^{-1}$, according to the Σ_{t+1}^{-1} update in Equation (8), $\Sigma_{t+1} = \Sigma_t$ due to the uniqueness of matrix inverses. This allows for the substitution of Σ_t for Σ_{t+1} in Equation (10), giving rise to the ODEs seen in Equation (3):

$$\frac{d\Sigma_t^{-1}}{dt} = \mathbb{E}_{q(\mathbf{x}; \alpha_t)} \left[\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \right] - \Sigma_t^{-1}, \quad \frac{d\boldsymbol{\mu}_t}{dt} = -\Sigma_t \mathbb{E}_{q(\mathbf{x}; \alpha_t)} \left[\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \right] \quad (11)$$

4 Implementation

This section discusses general key approximations, assumptions, and techniques used during the implementation of the particle flow using numerical integration methods to help with result replicability.

4.1 Initialization of mean ($\boldsymbol{\mu}_0$) and covariance (Σ_0)

The mean ($\boldsymbol{\mu}_t$) and covariance (Σ_t) are initialized using the prior mean and covariance: $\boldsymbol{\mu}_t = \hat{\mathbf{x}}$, $\Sigma_t = \mathbf{P}$.

4.2 Monte Carlo Approximation

As the Bayesian Estimation problem becomes more complex, the expectation expressions in Equations (2) and (3) become intractable due to requiring integration of high-dimensional vector functions. As a result, these expectations must be approximated, with the chosen method being Monte Carlo approximation:

$$\mathbb{E}_q[f(\mathbf{x})] \approx \frac{1}{N} \sum_{i=1}^N f(\mathbf{x}_i), \quad \mathbf{x}_i \sim q \quad (12)$$

The expectation approximation generally improves as the amount of samples N increases. Applying the Monte Carlo approximation to the VPFPPF parameter flows in Equations (2) and (3) gives the following flow expressions below:

$$\frac{d\Sigma_t^{-1}}{dt} = \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial^2 V(\mathbf{x}; \alpha_t)}{\partial \mathbf{x}^\top \partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i \sim q(\mathbf{x}; \alpha_t)} \right], \quad \frac{d\boldsymbol{\mu}_t}{dt} = -\Sigma_t \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial V(\mathbf{x}; \alpha_t)}{\partial \mathbf{x}^\top} \Big|_{\mathbf{x}=\mathbf{x}_i \sim q(\mathbf{x}; \alpha_t)} \right] \quad (13)$$

$$\frac{d\Sigma_t^{-1}}{dt} = \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} \Big|_{\mathbf{x}=\mathbf{x}_i \sim q(\mathbf{x}; \alpha_t)} \right] - \Sigma_t^{-1}, \quad \frac{d\boldsymbol{\mu}_t}{dt} = -\Sigma_t \left[\frac{1}{N} \sum_{i=1}^N \frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} \Big|_{\mathbf{x}=\mathbf{x}_i \sim q(\mathbf{x}; \alpha_t)} \right] \quad (14)$$

It should be noted that the approximated flows, Equations (13) and (14), will result in slight differences in the posterior's parameters, $\boldsymbol{\mu}_f$ and Σ_f , as the expectation, which allowed for the initial derivation of Equation (3) from (2), is now approximated. However, this difference is minimal with thousands of sampled particles, and this can be seen in Figure 1a, where the JAX method uses the Python-based JAX library's autodifferentiation to compute the derivatives to implement Equation (13), while the analytical method is based on Equation (14).

Monte Carlo Approximation is also applied to the derivative-free (Stein's Lemma) parameter flow. This new approximated flow is given by the following:

$$\frac{d\Sigma_t^{-1}}{dt} = \Sigma_t^{-1} \left[\frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top V(\mathbf{x}_i; \alpha_t) \right] \Sigma_t^{-1} - \Sigma_t^{-1} \left[\frac{1}{N} \sum_{i=1}^N V(\mathbf{x}_i; \alpha_t) \right] \quad (15)$$

$$\frac{d\boldsymbol{\mu}_t}{dt} = -\frac{1}{N} \sum_{i=1}^N [(\mathbf{x}_i - \boldsymbol{\mu})V(\mathbf{x}_i; \boldsymbol{\alpha}_t)]. \quad (16)$$

4.2.1 Random particle generation

For the Monte Carlo Approximation, to sample particles as $\mathbf{x}_i \sim q(\mathbf{x}|\boldsymbol{\alpha}_t)$, particles were generated according to

$$\mathbf{X} = \boldsymbol{\xi} \left[\sqrt{\boldsymbol{\Sigma}_t} \right]^\top + \boldsymbol{\mu}_t, \quad (17)$$

where $\boldsymbol{\xi} = [\boldsymbol{\epsilon}_1 \ \boldsymbol{\epsilon}_2 \ \cdots \ \boldsymbol{\epsilon}_N]^\top$ with $\boldsymbol{\epsilon}_i \in \mathbb{R}^d \sim \mathcal{N}(0, \mathbf{I})$, $\sqrt{\boldsymbol{\Sigma}_t}$ is the Cholesky decomposition of the covariance matrix, and $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_N]^\top \in \mathbb{R}^{N \times d}$. It should be noted that this method requires $\boldsymbol{\Sigma}$ to be positive-definite, which is a stronger requirement than the positive semi-definite condition that covariance matrices are defined to have. $N = 3000$ was the main choice for particle amount.

4.2.2 Unscented particles

Instead of sampling thousands of points randomly, $\mathbf{x}_i \sim q(\mathbf{x}; \boldsymbol{\alpha}_t)$, only $2d + 1$ sigma points can be deterministically chosen for \mathbf{x}_i using the Unscented Transform, where $\mathbf{x} \in \mathbb{R}^d$. Each sigma point $\mathbf{x}^{(i)}$ corresponds to a weight $v^{(i)}$, and both are found as follows, with hyperparameters $\alpha \in (0, 1]$ and $k > -d$ that determine the spread of the sigma points:

$$v^{(0)} = 1 - \frac{d}{\alpha^2(d+k)} < 1, \quad v^{(i)} = \frac{1 - v^{(0)}}{2d} \text{ for } i = 1, \dots, 2d \quad (18)$$

$$\mathbf{x}^{(0)} = \boldsymbol{\mu}, \quad \mathbf{x}^{(i)} = \boldsymbol{\mu} \pm \sqrt{\frac{d}{1 - v^{(0)}}} \left[\sqrt{\boldsymbol{\Sigma}} \right]_i \text{ for } i = 1, \dots, d \quad (19)$$

This method will be used along with the Monte-Carlo-approximated Stein's Lemma parameter flow to approximate the expectation. [6] recommends choosing $\alpha = 10^{-3}$ (or other similar small, nonzero positive value) and $k = 0$. This can be seen in Figure 1b.

4.3 Removing constant terms of $V(\mathbf{x}; \boldsymbol{\alpha}_t)$ for Stein's Lemma derivative-free implementation

When implementing the Stein's Lemma-based derivative-free flow for the function $V(\mathbf{x}; \boldsymbol{\alpha}_t)$, one may run into stability issues during the numerical integration process if there are constant terms in the differentiated function while using numerical approximations like Monte-Carlo. This issue, however, can be solved by removing any constant terms in the general function $f(\mathbf{x})$ in Stein's Lemma (for this work, $f(\mathbf{x}) = V(\mathbf{x}; \boldsymbol{\alpha}_t)$). This means that, as $f(\mathbf{x})$ can be written as $f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{c}$, where $\mathbf{x}, \mathbf{c} \in \mathbb{R}^n$ are deterministic, and $g(\mathbf{x})$ is a sum of terms entirely dependent on \mathbf{x} (e.g., $g(\mathbf{x}) = \mathbf{x}^2 + e^{\mathbf{x}} + \ln(\mathbf{x})$), one should set $\mathbf{c} = 0$.

Stein's lemma is known to work for any differentiable function $f(\mathbf{x})$ that satisfies $\mathbb{E}[\frac{\partial}{\partial \mathbf{x}^\top} f(\mathbf{x})] < \infty$. This includes functions that have affine components \mathbf{c} and comes from the properties of expectation.

By writing the general function $f(\mathbf{x})$ in the form $f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{c}$, it can be seen that the non-derivative direction of Stein's lemma correctly eliminates the constant term, and this re-derives Stein's Lemma for $g(\mathbf{x})$.

$$\begin{aligned}
\mathbb{E}_q \left[\frac{\partial(f(\mathbf{x}))}{\partial \mathbf{x}^\top} \right] &= \mathbb{E}_q \left[\frac{\partial(g(\mathbf{x}) + \mathbf{c})}{\partial \mathbf{x}^\top} \right] = \Sigma^{-1} \mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})(g(\mathbf{x}) + \mathbf{c})] \\
&= \Sigma^{-1} (\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})g(\mathbf{x})] + \mathbf{c} \mathbb{E}_q[\mathbf{x} - \boldsymbol{\mu}]) \\
&= \Sigma^{-1} (\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})g(\mathbf{x})] + \mathbf{c}(\boldsymbol{\mu} - \boldsymbol{\mu})) \\
&= \Sigma^{-1} \mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})g(\mathbf{x})]
\end{aligned}$$

However, because of the need to approximate the expectation, with this work using Monte Carlo Approximation, the behavior of the expectation above that eliminates the constant terms (i.e. $\mathbf{c} \mathbb{E}_q[\mathbf{x} - \boldsymbol{\mu}] = \mathbf{c}(\boldsymbol{\mu} - \boldsymbol{\mu}) = 0$) is also approximated, and this can lead to instability during the iterative process, as $\mathbb{E}_q \left[\frac{\partial(f(\mathbf{x}))}{\partial \mathbf{x}^\top} \right] = \Sigma^{-1} \mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})g(\mathbf{x})]$ is no longer necessarily true.

If $f(\mathbf{x})$ has an affine component, the term $\mathbf{c} \mathbb{E}_q[\mathbf{x} - \boldsymbol{\mu}] \approx \frac{\mathbf{c}}{N} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu}) \neq 0$ for low N , as $\mathbb{E}_q[\mathbf{x}] \approx \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i) \neq \boldsymbol{\mu}$, reducing the accuracy of the approximation of $\mathbb{E}_q \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right]$. Removing any constant terms from $f(\mathbf{x})$ will still result in a function that will approximate $\mathbb{E}_q \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right]$, as the derivative is invariant to different constant terms. However, this now enforces the condition of $\mathbf{c} \mathbb{E}_q[\mathbf{x} - \boldsymbol{\mu}] \approx \frac{\mathbf{c}}{N} \sum_{i=1}^N (\mathbf{x}_i - \boldsymbol{\mu}) = 0$, as $\mathbf{c} = 0$, resulting in a better approximation of $\mathbb{E}_q \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^\top} \right]$ and less instability during the numerical integration process.

5 Evaluation

In this section, a Nonlinear Observation Likelihood Bayesian Estimation problem is formulated, and the many different solution methods for implementing the VPFPF parameter flows, analytical, JAX-based autodifferentiation, and derivative-free (Stein's Lemma), are discussed and compared.

5.1 Considered problems

Nonlinear Observation Likelihood A problem with a nonlinear observation model will be considered, where the prior follows a single Gaussian distribution given by $p(\mathbf{x}) = p_{\mathcal{N}}(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{P})$, with

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 5.5 & -1.5 \\ -1.5 & 5.5 \end{bmatrix},$$

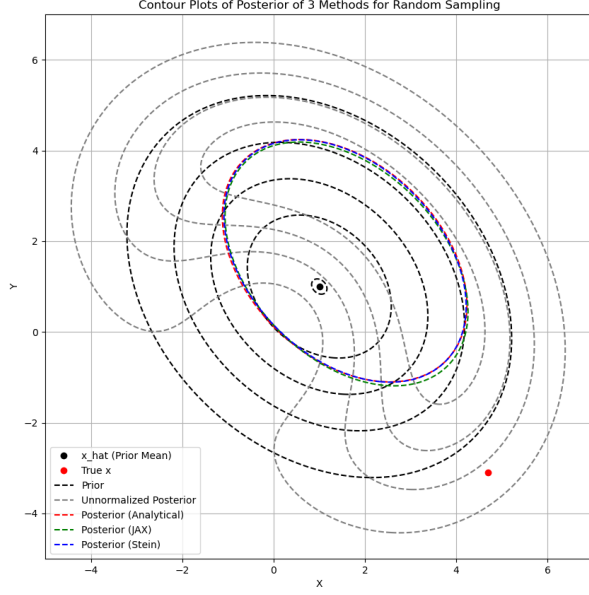
and Gaussian likelihood function $p(\mathbf{z}|\mathbf{x}) = p_{\mathcal{N}}(\mathbf{z}; H(\mathbf{x}), R)$, with

$$H(\mathbf{x}) = \|\mathbf{x}\|_2, \quad R = 2, \quad \mathbf{x}^* = \begin{bmatrix} 4.7 \\ -3.1 \end{bmatrix}.$$

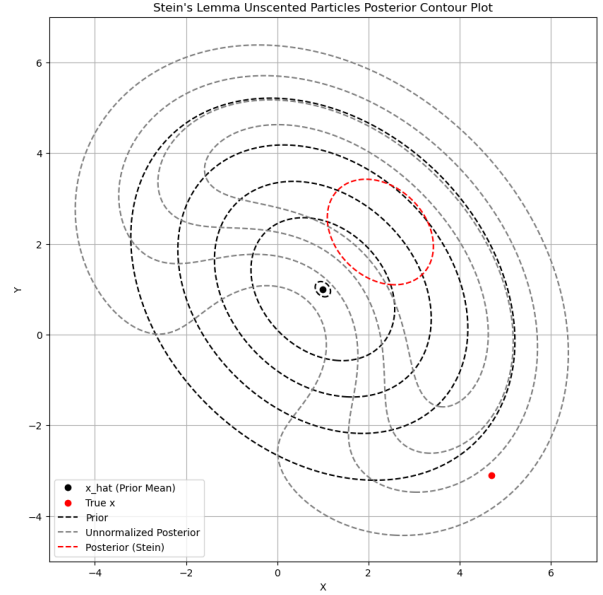
\mathbf{x}^* is the true state used to generate an observation, $\mathbf{z} = H(\mathbf{x}) + \sqrt{R}\boldsymbol{\epsilon}$, for $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$ and $\boldsymbol{\epsilon}\mathbf{x}$. Also, the assumption that an observation \mathbf{z} given by the following will be made:

$$\mathbf{z} = H(\mathbf{x}^*) = \left\| \begin{bmatrix} 4.7 \\ -3.1 \end{bmatrix} \right\|_2$$

The goal is to find the posterior $p(\mathbf{x}|\mathbf{z})$ given the prior and likelihood above, and the main method used was Gaussian Variational Inference.



(a) The analytical posterior (red), JAX posterior (green), and derivative-free (Stein's Lemma) posterior (blue) all agree.



(b) Stein's Lemma derivative-free implementation with Unscented Particles corresponding to $\alpha = 10^{-3}$ and $k = 0$.

Figure 1: The contour plots of the prior and different posterior solutions for the nonlinear observation model problem are plotted for comparison. The prior (black) and unnormalized posterior (gray) are plotted for reference, and all of the approximated posteriors (analytical, JAX, Stein's Lemma, and Stein's Lemma Unscented) are plotted with a covariance contour corresponding to one Mahalanobis distance. The unscented particle result (right) is closer than the results with randomly generated particles (left) to the result seen in [7] for the same nonlinear observation model

5.2 Solution methods

A comparison of the final posteriors of the three methods can be seen in Figure 1a. All three methods return very similar, if not identical, posteriors.

5.2.1 Analytically-computed derivatives solution

As this problem has $\mathbf{x} \in \mathbb{R}^2$, analytically calculating the gradients and Hessians in the VPFPPF is possible. Due to the simpler gradient and Hessian expressions in Equation (14), the following expressions for the gradient and Hessian will be derived for it:

$$\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} = \frac{\mathbf{x}}{R^2} - \frac{\mathbf{z}\mathbf{x}}{R^2 \|\mathbf{x}\|_2} + \mathbf{P}^{-1}(\mathbf{x} - \hat{\mathbf{x}}), \quad \frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} = \frac{\mathbf{I}}{R^2} - \frac{\mathbf{z}}{R^2} \left(\frac{\mathbf{I}}{\|\mathbf{x}\|_2} - \frac{\mathbf{x}\mathbf{x}^\top}{\|\mathbf{x}\|_2^3} \right) + \mathbf{P}^{-1} \quad (20)$$

Applying these to the approximated parameter flows in Equation (14) results in the following, for $\mathbf{x}_i \sim q(\mathbf{x}|\alpha_t)$:

$$\frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \frac{1}{N} \sum_{i=1}^N \left[\frac{\mathbf{I}}{R^2} - \frac{\mathbf{z}}{R^2} \left(\frac{\mathbf{I}}{\|\mathbf{x}_i\|_2} - \frac{\mathbf{x}_i\mathbf{x}_i^\top}{\|\mathbf{x}_i\|_2^3} \right) + \mathbf{P}^{-1} \right] - \boldsymbol{\Sigma}_t^{-1} \quad (21)$$

$$\frac{d\boldsymbol{\mu}_t}{dt} = -\frac{\boldsymbol{\Sigma}_t}{N} \sum_{i=1}^N \left[\frac{\mathbf{x}_i}{R^2} - \frac{\mathbf{z}\mathbf{x}_i}{R^2 \|\mathbf{x}_i\|_2} + \mathbf{P}^{-1}(\mathbf{x}_i - \hat{\mathbf{x}}) \right] \quad (22)$$

Derivation First, by taking into account the given information for the Nonlinear Observation Model problem formulated in Section 5.1, $\phi(\mathbf{x})$ can be written as follows:

$$\begin{aligned}
\phi(\mathbf{x}) &= -\ln p(\mathbf{x}, \mathbf{z}) \\
&= -\ln[p(\mathbf{z}|\mathbf{x})p(\mathbf{x})] \\
&= -\ln \left[\frac{1}{\sqrt{2\pi R^2}} e^{-\frac{1}{2R^2}(\mathbf{z}-\|\mathbf{x}\|_2)^2} \frac{1}{\sqrt{(2\pi)^N |\mathbf{P}|}} e^{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}})} \right] \\
&= \frac{1}{2} \ln(2\pi R^2) + \frac{1}{2R^2}(\mathbf{z}-\|\mathbf{x}\|_2)^2 \\
&\quad + \frac{1}{2} \ln((2\pi)^N |\mathbf{P}|) + \frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}})
\end{aligned} \tag{23}$$

Again, Equation (14) is used due to its simpler derivatives, which results in the following steps to find the gradient of $\phi(\mathbf{x})$:

$$\begin{aligned}
\frac{\partial \phi(\mathbf{x})}{\partial \mathbf{x}^\top} &= \frac{1}{R^2}(\mathbf{z}-\|\mathbf{x}\|_2) \left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2} \right) + \frac{1}{2}(2\mathbf{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}})) \\
&= \frac{(\|\mathbf{x}\|_2 - \mathbf{z})\mathbf{x}}{R^2\|\mathbf{x}\|_2} + \mathbf{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}}) \\
&= \frac{\mathbf{x}}{R^2} - \frac{\mathbf{z}\mathbf{x}}{R^2\|\mathbf{x}\|_2} + \mathbf{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}})
\end{aligned}$$

Taking the derivative of the above expression results in the hessian, according to the following steps:

$$\begin{aligned}
\frac{\partial^2 \phi(\mathbf{x})}{\partial \mathbf{x}^\top \partial \mathbf{x}} &= \frac{\mathbf{I}}{R^2} - \frac{\mathbf{z}}{R^2} \left(\frac{\mathbf{I}}{\|\mathbf{x}\|_2} - \frac{\mathbf{x}\mathbf{x}^\top}{\|\mathbf{x}\|_2^3} \right)^\top + \mathbf{P}^{-1} \\
&= \frac{\mathbf{I}}{R^2} - \frac{\mathbf{z}}{R^2} \left(\frac{\mathbf{I}}{\|\mathbf{x}\|_2} - \frac{\mathbf{x}\mathbf{x}^\top}{\|\mathbf{x}\|_2^3} \right) + \mathbf{P}^{-1}
\end{aligned}$$

Then, substituting these into the update equations gives the following final equations for the analytical implementation of the VPFPF parameter flows:

$$\frac{d\boldsymbol{\Sigma}_t^{-1}}{dt} = \frac{1}{N} \sum_{i=1}^N \left[\frac{\mathbf{I}}{R^2} - \frac{\mathbf{z}}{R^2} \left(\frac{\mathbf{I}}{\|\mathbf{x}_i\|_2} - \frac{\mathbf{x}_i\mathbf{x}_i^\top}{\|\mathbf{x}_i\|_2^3} \right) + \mathbf{P}^{-1} \right] - \boldsymbol{\Sigma}_t^{-1} \tag{24}$$

$$\frac{d\boldsymbol{\mu}_t}{dt} = -\frac{\boldsymbol{\Sigma}_t}{N} \sum_{i=1}^N \left[\frac{\mathbf{x}_i}{R^2} - \frac{\mathbf{z}\mathbf{x}_i}{R^2\|\mathbf{x}_i\|_2} + \mathbf{P}^{-1}(\mathbf{x}_i - \hat{\mathbf{x}}) \right] \tag{25}$$

5.2.2 Autodifferentiated (JAX) solution

Instead of calculating the gradient and hessian by hand, like in Equation (20), a library with autodifferentiation like Python's JAX can be used calculate the gradient and hessian, using *jax.grad*($V(\mathbf{x}, \boldsymbol{\alpha}_t)$) and *jax.hess*($V(\mathbf{x}, \boldsymbol{\alpha}_t)$) respectively. The JAX posterior can be seen in Figure 1a as the green contour.

5.2.3 Derivative-free (Stein’s Lemma) Solution

The derivative-free solution is implemented using both random-sampled particles and $2d + 1$ unscented particles. The random-sampled particles results in a posterior agreeing with the analytical and JAX solutions above, seen in Figure 1a, while the unscented particles (Figure 1b) give the closest results to those seen in [7].

5.3 Unnormalized posterior

The true normalized posterior cannot be found due to the intractability of $p(\mathbf{z})$, but the unnormalized posterior $p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$ can be, which is a good reference for the overall shape of the true posterior. In code, this can be found by element-wise multiplying the output points of the prior and likelihood together when plotting. This is the gray contour in Figures 1a and 1b.

6 Conclusion

This work showed the agreement between implementations of the VPFPP parameters flows using analytically-computed, autodifferentiated, and Stein’s Lemma-based derivatives for thousands of randomly generated particles. Next, this work showed how the use of unscented particles can produce better results with only a few deterministically-chosen particles, using the results from [7] as a baseline. Finally, to simplify and improve the implementations, a modified continuous formulation and a discrete formulation of the VPFPP parameter flows were unveiled, and the importance of removing affine terms in the test function $f(\mathbf{x})$ to improve numerical stability when approximating Stein’s Lemma was shown. Future work will explore the application of the VPFPP parameter flows to estimation problems with evolving observations and states, factoring in new variables like a changing robot position that generates new observations.

References

- [1] T. D. Barfoot, J. R. Forbes, and D. J. Yoon. Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *The International Journal of Robotics Research*, 39(13):1473–1502, 2020.
- [2] T. Bayes. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418, 1763.
- [3] F. Daum and J. Huang. Particle flow for nonlinear filters. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5920–5923, 2011.
- [4] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, Nov. 1999.
- [5] C. M. Stein. Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9(6):1135 – 1151, 1981.
- [6] E. 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)*, pages 153–158, 2000.
- [7] Y. Yi, J. Cortés, and N. Atanasov. Variational formulation of the particle flow particle filter. *arXiv preprint arXiv:2505.04007*, 2025.