Computational Foundation for Optimal Transport Ensemble Kalman Filter

Hasan Emin Horata

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics and Astronautics

University of Washington
2023

Committee:
Amirhossein Taghvaei
Bamdad Hosseini

Program Authorized to Offer Degree:
Aeronautics and Astronautics

University of Washington

Abstract

Computational Foundation for Optimal Transport Ensemble Kalman Filter

Hasan Emin Horata

Chair of the Supervisory Committee:
Professor Amirhossein Taghvaei

Aeronautics and Astronautics

This thesis is concerned with novel filtering algorithms that are constructed based on the recently introduced optimal transportation formulation of the Bayes' formula. The problem involves solving a stochastic optimization problem to find the optimal map that transports the prior distribution to the posterior distribution at each time-step. An insightful case is obtained when the functions are restricted to be quadratic, leading to a variation of ensemble Kalman filter (EnKF), called optimal transport EnKF(OT-EnKF). The thesis includes development of computational algorithms for OT-EnKF and performing numerical experiments for evaluation and comparison with the original EnKF algorithm.

Acknowledgements

Firstly, I would like to thank Dr. Amirhossein Taghvaei. I remember when we started a year ago on this project that I was terrified of statistics. However, with your mentor and guidance, not only did I get over that fear, but you taught me to think deeper about my problems, to go from 'What is the solution?' to 'Why is that the solution?' This thinking and curiosity will help me well in my life, and again I thank you for that. I'd also like to thank Dr. Bamdad Hosseini for being a part of the committee and for his support as well. And last but not least, I want to thank my family and my friends for showing me support and for making my time at the University of Washington exciting.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

Control systems has developed rapidly in the past century due to the advancement of communication and computing technology, resulting in various applications such as the landing of rockets, communicating between satellites, and developing efficient aircraft. A safe and reliable operation of a control system requires an accurate estimate of the state of the system and uncertainty around it. This is formally called estimation and uncertainty quantification (UQ).

To achieve this, the system relies on a model for the dynamics of the system and real-time data collected from sensors such as accelerometer or odometer. Unfortunately, there is no such thing as a perfect model or sensor. There are uncertainties associated with modelling and noise in the sensory data that makes the estimation and UQ problem challenging. This is where Rudolf Kalman enters the story. Kalman developed his famous technique called the Kalman filter (KF) [16, 17] which allowed engineers and scientists to extract useful information from noisy sensory signals. Ever since, it has been used in many different industries, with the most famous example being the Apollo program, helping the first men land on the moon [11].

However, the application of KF to large-scale problems is computationally limited due to the following two reasons:

1. Application of the KF requires an explicit and accurate analytical model for the system dynamics and observation likelihood. Such analytical models are often not available in high-dimensional applications that involve numerical solutions to partial differential equations.

2. Implementation of KF in high-dimensional setting becomes computationally expensive due to propagation of error covariance matrix that scales with the square of the state dimension.

Such limitations motivated development of sequential Monte-Carlo algorithms to KF, that are genrally called Ensemble Kalman filter (EnKF) [10, 19, 2, 6]. Comapred to KF, EnKF only requires a simulator for the dynamic model and scales well with the problem dimension[13]. As a result, it became

a workhorse for weather prediction applications in atmospheric science, where the dimension of the state is of order 10^8, and it has has been operational at the National Center for Environmental Prediction [29] and the Canadian Meteorological Centre [14].

There are several distinct types of the EnKF algorithm, with the most well-known ones being the (i) EnKF with perturbed observation [9]; and (ii) The square root EnKF [28]. The details for these algorithms appear in [20, Ch. 6-7]. An excellent recent survey on this topic appears in [6]. Error analysis of the EnKF algorithm is an active area of research with comprehensive review of recent developments in [4].

The objective of this thesis is to study the EnKF algorithm from both the design and computational perspectives. In particular, the objective is to explore the application of the recently introduced optimal transport formulation of the Bayes' law for construction of, what we call, the optimal transport EnKF (OT-ENKF) [23]. The outline of the thesis, among with its contributions, is as follows:

- Chapter 2 includes background on the estimation problem. It introduces the class of linear Gaussian dynamical systems and the Kalman filter algorithm, along with an illustration of it with a numerical example.

- Chapter 3 introduces the EnKF algorithm first from a mean-field prospective, and then its particle approximation. The algorithm is illustrated with the aid of a numerical example. The numerical results include comparison with the Kalman filter, and numerical error analysis as the number of particles grow.

- Chapter 4 introduces the new variation of the EnKF algorithm using the optimal transportation methodology. This is called the OT-EnKF method. Much like chapter 3, it first derives the algorithm from a mean-field perspective, and then uses the particle process to calculate the empirical cost. Two different OT-EnKF techniques are derived.

- Chapter 5 shows the results of the experiments ran to test the new OT-EnKF techniques. First it is shown how the parameters converge to their optimal values. Then, it is seen how the mean square error compared between the Kalman filter and the EnKF and the OT-EnKF techniques vary based on some changing factors. Lastly, we compare the EnKF and OT-EnKF in a linear dynamical system to observe how they perform.

- Chapter 6 ends the thesis with a summary of what has been learned. It further provides recommendation for future work.

Chapter 2

Background

2.1 Estimation problem

We start by considering a simple estimation problem. Assume $X \in \mathbb{R}^n$ is the hidden state of a system, and $Y \in \mathbb{R}^m$ is the observation signal. Assume X and Y are both random variables with the joint probability distribution P_{XY}. The problem is to estimate X based on the observation Y. To make this problem well-defined, we use the mean-squared-error (mse) criterion

$$\min_{\hat{X} \in \sigma(Y)} \mathbb{E}[\|X - \hat{X}\|^2] \tag{2.1}$$

where $\hat{X}$ represents the estimate and $\sigma(Y)$ is the set of random variables that are measurable with respect to Y (in simple terms, it means that $\hat{X}$ is a function of Y). This is known as the mse estimation problem. The minimizer can be viewed as the projection of the random variable X to the linear subspace of Y-measurable random variables $\sigma(Y)$ [15]. It is equal to, or can be used as a definition for, the conditional expectation

$$\mathbb{E}[X|Y] = \int x P_{X|Y}(x|Y) dx$$

where $P_{X|Y}$ is the conditional distribution of X given Y. The fact that $\mathbb{E}[X|Y]$ is the minimizer can be seen by using the decomposition:

$$\begin{aligned}
\mathbb{E}[\|X - \hat{X}\|^2] &= \mathbb{E}[\|(X - \mathbb{E}[X|Y]) + (\mathbb{E}[X|Y] - \hat{X})\|^2] \\
&= \mathbb{E}[\|X - \mathbb{E}[X|Y]\|^2] + \mathbb{E}[\|\mathbb{E}[X|Y] - \hat{X}\|^2] \\
&\quad + 2\mathbb{E}[(X - \mathbb{E}[X|Y])(\mathbb{E}[X|Y] - \hat{X})]
\end{aligned}$$

and observing that the last term is zero due to the tower-property of conditional expectations:

$$\begin{aligned}\mathbb{E}[(X-\mathbb{E}[X|Y])(\mathbb{E}[X|Y]-\hat{X})] &= \mathbb{E}[\mathbb{E}[(X-\mathbb{E}[X|Y])(\mathbb{E}[X|Y]-\hat{X})|Y]] \\ &= \mathbb{E}[(\mathbb{E}[X|Y]-\hat{X})\mathbb{E}[(X-\mathbb{E}[X|Y])|Y]] \\ &= \mathbb{E}[(\mathbb{E}[X|Y]-\hat{X})(\mathbb{E}[X|Y]-\mathbb{E}[X|Y])] \\ &= 0\end{aligned}$$

As a result,

$$\mathbb{E}[\|X-\hat{X}\|^2] = \mathbb{E}[\|X-\mathbb{E}[X|Y]\|^2] + \mathbb{E}[\|\mathbb{E}[X|Y]-\hat{X}\|^2]$$

minimized by $\hat{X} = \mathbb{E}[X|Y]$. This is a valid minimizer because the conditional expectation $\mathbb{E}[X|Y]$ is measurable with respect to Y, thus belongs to $\sigma(Y)$. Given the best mse estimator, we define the error covariance matrix according to

$$\Sigma = \mathbb{E}[(X-\mathbb{E}[X|Y])(X-\mathbb{E}[X|Y])^\top]$$

2.2 Linear estimation

Now, consider the estimation problem in the setting that the joint distribution P_{XY} is Gaussian. Suppose the mean vector and the covariance matrix of this Gaussian distribution is given by:

$$\mathcal{N}(\begin{bmatrix} m_X \\ m_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}$$

where

$$\Sigma_{Z_1 Z_2} := \mathbb{E}\left[(Z_1-\mathbb{E}[Z_1])(Z_2-\mathbb{E}[Z_2])^\top\right]$$

for any two random variables Z_1 and Z_2.

It is known that, in the Gaussian case, the optimal estimator, i.e. the conditional expectation, is an affine function of Y [12]

$$\mathbb{E}[X|Y] = KY + b$$

where K is a $n \times m$ matrix and b is a n-dimensional vector. Upon using the minimum mse estimation property of conditional expectation, the optimal values for the parameters K and b are obtained by solving

$$\min_{K,b} \mathbb{E}[\|X-KY-b\|^2]. \tag{2.2}$$

The solution is obtained by writing the first-order optimality conditions. In particular, the gradient of the objective function $J(K,b) := \mathbb{E}[\|X-KY-b\|^2]$ with respect to K and b are given by:

$$\begin{aligned}\nabla_b J(K,b) &= -2\mathbb{E}[X-KY-b] \\ \nabla_K J(K,b) &= -2\mathbb{E}[(X-KY-b)Y^\top]\end{aligned}$$

Solving (K, b) for $\nabla_K J(K, b) = 0$ and $\nabla_b J(K, b) = 0$ yields

$$K := \Sigma_{XY}\Sigma_{YY}^{-1} \tag{2.3a}$$
$$b := m_X - Km_Y \tag{2.3b}$$

Upon using the optimal estimator, the error covariance matrix is equal to

$$\mathbb{E}[(X - \mathbb{E}[X|Y])(X - \mathbb{E}[X|Y])^\top] = \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}$$

We use these formulas to build the Kalman filter in the next section.

2.3 Kalman filter

The Kalman filter is designed to solve the estimation problem in a dynamic setting [16, 15]. Consider a linear dynamical system system governed by the equations

$$X_t = AX_{t-1} + \sigma_v V_{t-1} \tag{2.4a}$$
$$Y_t = CX_t + \sigma_w W_t, \tag{2.4b}$$

for $t = 1, 2, 3, \ldots$, where $X_t \in \mathbb{R}^n$ is the state of the system, $Y_t \in \mathbb{R}^m$ is the observation signal, V_t and W_t are standard Gaussian random variables of dimensions m and n respectively, and σ_v and σ_w are $n \times n$ and $m \times m$ matrices. The initial state X_0 is a Gaussian random vector with mean m_0 and covariance matrix Σ_0, which is denoted by $X_0 \sim N(m_0, \Sigma_0)$.

The Kalman filter is a sequential algorithm that gives the best mse estimate of X_t given history of the observation $\mathcal{Y}_t := \{Y_1, \ldots, Y_t\}$, for all t. From the discussions of Section 2.1, we know that the optimal mse estimate is equal to the conditional expectation $\mathbb{E}[X_t|\mathcal{Y}_t]$. We will use m_t to denote the conditional mean and Σ_t to denote the conditional error covariance:

$$m_t := \mathbb{E}[X_t|\mathcal{Y}_t] = \arg\min_{\hat{X} \in \sigma(\mathcal{Y}_t)} \mathbb{E}[\|X_t - \hat{X}\|^2],$$
$$\Sigma_t := \mathbb{E}[(X_t - m_t)(X_t - m_t)^\top]$$

The Kalman filter algorithm provides the update laws for the mean m_t and the covariance matrix Σ_t:

$$\text{(prediction)} \qquad m_t^- = Am_{t-1}, \quad \Sigma_t^- = A\Sigma_{t-1}A^\top + \Sigma_v, \tag{2.5a}$$
$$\text{(update)} \quad m_t = m_t^- + K_t(Y_t - Cm_t^-), \quad \Sigma_t = (I - K_tC)\Sigma_t^- \tag{2.5b}$$

where $\Sigma_v = \sigma_v\sigma_v^\top$, $K_t = \Sigma_t^- C^\top S_t$, and $S_t = (C\Sigma_t^- C^\top + \sigma_w\sigma_w^\top)^{-1}$. The first equation is an update according to the dynamic model. It provides a prediction for the state X_t. The second equation is an update due to conditioning with the observation Y_t. It has the same form as the optimal linear estimator formula obtained in Section 2.2. The Kalman filter algorithm is also presented in Table 1.

Algorithm 1 Kalman Filter

Input: $\{Y_1, Y_2, \ldots, Y_T\}$, m_0, Σ_0, model parameters $(A, C, \Sigma_v, \Sigma_w)$
Output: estimate and error covariance matrix $(m_t, \Sigma_t)_{t=1}^T$
for $t = 1$ to T **do**
  Prediction step:
    $m_t^- = Am_{t-1}$
    $\Sigma_t^- = A\Sigma_{t-1}A^\top + \Sigma_v$
  Update step:
    $K_t = \Sigma_t^- C^\top (C\Sigma_t^- C^\top + \Sigma_w)^{-1}$
    $m_t = m_t^- + K_t(Y_t - Cm_t^-)$
    $\Sigma_t = (I - K_t C)\Sigma_t^-$
end for

2.4 Numerical example

Consider a single mass-spring system that is modelled with the second-order differential equation:

$$\ddot{z}(t) = -\omega^2 z(t) + u(t)$$

where $z(t)$ is the position, ω is the frequency, and $u(t)$ represent random disturbances such that $\mathbb{E}u(t) = 0$ and $\mathbb{E}u(t)u(s) = 0.1\delta_{t=s}$. Converting to the state-space form with $X(t) = [X_1(t), X_2(t)] = [Z(t), \dot{Z}(t)]$ and time discretization, the dynamics model is of the form (2.4a) with the following values for the model parameters:

$$A = \begin{bmatrix} \cos(\omega\Delta t) & -\sin(\omega\Delta t) \\ \sin(\omega\Delta t) & \cos(\omega\Delta t) \end{bmatrix}, \quad \sigma_B = \begin{bmatrix} 0.0 & 0 \\ 0 & 0.1\Delta t \end{bmatrix},$$

We also assume access to noisy measurements of the position which is modelled in the form of (2.4b) with

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \sigma_W = \frac{0.1}{\Delta t}.$$

The initial state is assumed to have a standard Gaussian distribution with zero mean and identity covariance. A sample trajectory of the states of the system $X_t = [X_t(1), X_t(2)]$ and the observation signal Y_t is depicted in Figure 2.1 for $\omega = 2\pi$ and $\Delta t = 0.1$. We simulate the Kalman filter algorithm 1 for this realization of the observation signal. The resulting Kalman filter estimate $m_t = [m_t(1), m_t(2)]$ is depicted against the true values of the state in Figure 2.1. It is observed that the error between the true state and the Kalman filter estimate decreases with time. In order to quantify this, define the mean-squared-error in estimating the state with the Kalman filter (KF) according to:

$$\text{mse}_X^{\text{KF}}(t) := \mathbb{E}\|X_t - m_t\|^2$$

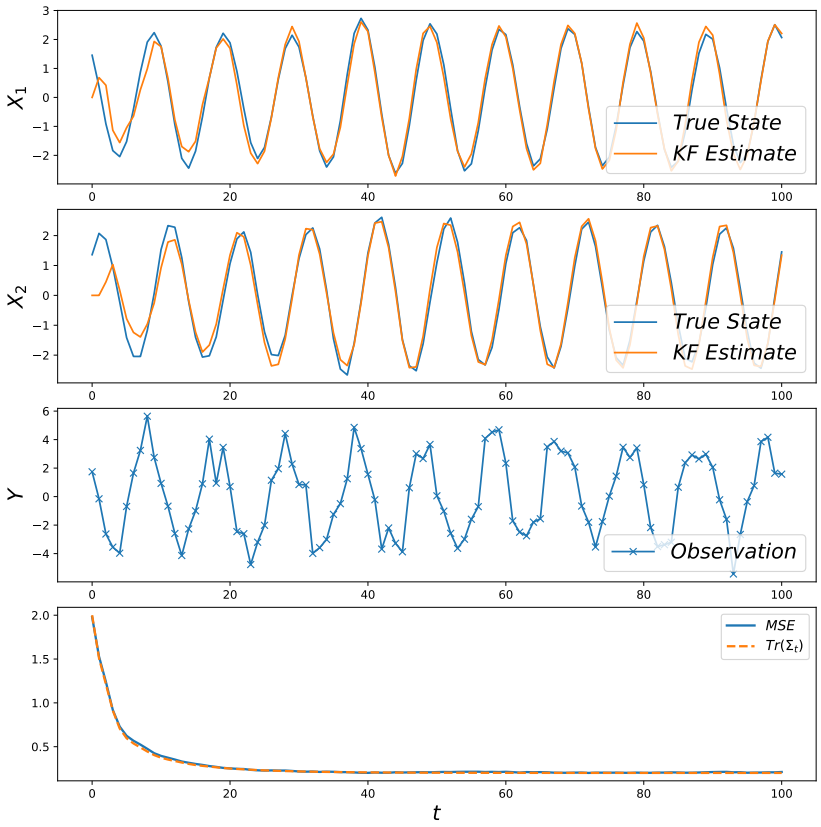

Figure 2.1: The first three graphs illustrate the sample trajectory of the states, observation, and Kalman-filter estimates, for the spring-mass example 2.4. It also compares the average mean square error between the true state and the Kalman-fitler estimate plotted against its expected value $\mathrm{Tr}(\Sigma_t)$.

We evalaute the mse by doing empirical average over 1000 independent Monte-Carlo simulations. The result is depicted in Figure 2.1. It is observed that the average mse is close it its expectation

$$\begin{aligned}\mathbb{E}[\|X_t - m_t\|^2] &= \mathbb{E}\left[\mathrm{Tr}\left[(X_t - m_t)(X_t - m_t)^\top\right]\right] \\ &= \mathrm{Tr}\left[\mathbb{E}\left[(X_t - m_t)(X_t - m_t)^\top\right]\right] \\ &= \mathrm{Tr}(\Sigma_t).\end{aligned}$$

Chapter 3

Ensemble Kalman Filter

The Ensemble Kalman filter(EnKF) can be viewed as a Monte-Carlo approximation of the Kalman filter [10, 19, 2, 6]. The construction of the EnKF can be done in two stages:

- Construct a mean-field process $\bar{X}_t$ such that it has the same distribution as the filter posterior distribution

$$\bar{X}_t \sim \pi_t.$$

- Approximate the mean-field process by simulating a system of N controlled interacting particles $\{X_1^i, \dots, X_t^N\}_{i=1}^N$. Then, the posterior distribution is approximated with the empirical distribution formed by the particles:

$$\pi_t \approx \frac{1}{N} \sum_{i=1}^{N} \delta_{X_t^i}.$$

We present these two stages for the linear model (2.4).

3.1 Mean-field process

The mean-field process $\bar{X}_t \in \mathbb{R}^n$ is constructed such that its distribution is equal to the posterior distribution π_t. For the linear system (2.4), the posterior is Gaussian $N(m_t, \Sigma_t)$ with the mean m_t and covairance matrix Σ_t given by the Kalman filter update equations (2.5). As we will later discuss in Chapter 4, there are infinitely many mean-field processes that satisfy this condition. A simple one is given by the so-called EnKF with perturbed observation [10]:

$$\text{(prediction)} \quad \bar{X}_t^- = A\bar{X}_{t-1} + \sigma_v \bar{V}_{t-1} \tag{3.1a}$$

$$\text{(update)} \quad \bar{X}_t = \bar{X}_t^- + \bar{K}_t(Y_t - \bar{Y}_t^-), \quad \bar{Y}_t^- = C\bar{X}_t^- + \sigma_w \bar{W}_t \tag{3.1b}$$

where $\bar{X}_0 \sim \mathcal{N}(m_0, \Sigma_0)$, $\bar{V}_t$ and $\bar{W}_t$ are independent copies of the process and measurement noise V_t and W_t in (2.4), and the gain

$$\bar{K}_t = \mathrm{Cov}(\bar{X}_t^-, \bar{Y}_t^- | \mathcal{Y}_t) \mathrm{Cov}(\bar{Y}_t^-, \bar{Y}_t^- | \mathcal{Y}_t)^{-1} \tag{3.2}$$

where we introduced the notation $\mathrm{Cov}(Z_1, Z_2)$ for the covariance matrix between two random variables

$$\mathrm{Cov}(Z_1, Z_2 | \mathcal{Y}) := \mathbb{E}[(Z_1 - \mathbb{E}[Z_1])(Z_2 - \mathbb{E}Z_2)^\top | \mathcal{Y}]$$

Define the mean and the covariance of the mean-field process according to

$$\begin{aligned} \bar{m}_t &:= \mathbb{E}[\bar{X}_t | \mathcal{Y}_t], \\ \bar{\Sigma}_t &:= \mathbb{E}[(\bar{X}_t - \bar{m}_t)(\bar{X}_t - \bar{m}_t)^\top | \mathcal{Y}_t] \end{aligned}$$

Then, we can show that the mean and covariance of the mean-field process, are equal to the mean and covariance given by the Kalman filter, i.e.

$$\bar{m}_t = m_t, \quad \bar{\Sigma}_t = \Sigma_t$$

In order to see this, take the conditional expectation of the equation for $\bar{X}_t$ in (3.1) to obtain:

$$\begin{aligned} \bar{m}_t = \mathbb{E}[\bar{X}_t | \mathcal{Y}_t] &= \mathbb{E}[\bar{X}_t^- | \mathcal{Y}_t] + K_t \mathbb{E}[Y_t - \bar{Y}_t^- | \mathcal{Y}_t] \\ &= \bar{m}_t^- + \bar{K}_t (Y_t - C \bar{m}_t^-) \end{aligned}$$

where we defined $\bar{m}_t^- := \mathbb{E}[\bar{X}_t^- | \mathcal{Y}_t]$, and used the fact that $\bar{W}_t$ is independent of $\mathcal{Y}_t$. The formula for $\bar{m}_t^-$ is obtained by taking the conditional expectation of the equation for $\bar{X}_t^-$ in (3.1)

$$\begin{aligned} \bar{m}_t^- &= \mathbb{E}[\bar{X}_t^- | \mathcal{Y}_t] = \mathbb{E}[A \bar{X}_{t-1} + \sigma_v \bar{V}_{t-1} | \mathcal{Y}_t] \\ &= A \mathbb{E}[\bar{X}_{t-1} | \mathcal{Y}_{t-1}] = A \bar{m}_{t-1} \end{aligned}$$

where we used the fact that $\bar{X}_{t-1}$ and $\bar{V}_{t-1}$ are independent of Y_t. As a result, the update equations for the conditional mean $\bar{m}_t$ are similar to the update equations for the conditional mean m_t given by the Kalman filter (2.5).

$$\bar{m}_t^- = A \bar{m}_{t-1}, \tag{3.3a}$$

$$\bar{m}_t = \bar{m}_t^- + \bar{K}_t (Y_t - C \bar{m}_t^-) \tag{3.3b}$$

Therefore, with the same initial value $\bar{m}_0 = m_0$, we have $m_t = \bar{m}_t$ for all t.

The fact that $\bar{\Sigma}_t = \Sigma_t$ follows from similar steps. Define the error process $e_t := \bar{X}_t - \bar{m}_t$. Subtracting the update equations for $\bar{X}_t$ in (3.1) from the update equations for $\bar{m}_t$ yields

$$\begin{aligned} e_t^- &= A e_{t-1} + \sigma_v \bar{V}_{t-1} \\ e_t &= (I - \bar{K}_t C) e_t^- - \bar{K}_t \sigma_w \bar{W}_t \end{aligned}$$

An update equation for $\bar{\Sigma}_t = [e_t e_t^\top | \mathcal{Y}_t]$ is obtained by writing the Lyapunov equations:

$$\begin{aligned}
\bar{\Sigma}_t^- &= \mathbb{E}[e_t^-(e_t^-)^\top | \mathcal{Y}_t] = A\bar{\Sigma}_{t-1}A^\top + \Sigma_v \\
\bar{\Sigma}_t &= \mathbb{E}[e_t e_t^\top | \mathcal{Y}_t] = (I - \bar{K}_t C)\bar{\Sigma}_t^- (I - \bar{K}_t C)^\top + \bar{K}_t \Sigma_w \bar{K}_t^\top \\
&= \Sigma_t^- - \bar{K}_t C\bar{\Sigma}_t^- - \Sigma_t^- C^\top \bar{K}_t^\top + \bar{K}_t(C\Sigma_t^- C^\top + \Sigma_w)\bar{K}_t^\top \\
&= \Sigma_t^- - \bar{K}_t C\Sigma_t^-
\end{aligned}$$

where we used the fact that

$$\begin{aligned}
\bar{K}_t &= \mathrm{Cov}(\bar{X}_t^-, \bar{Y}_t^- | \mathcal{Y}_t)\mathrm{Cov}(\bar{Y}_t^-, \bar{Y}_t^- | \mathcal{Y}_t)^{-1} \\
&= \bar{\Sigma}_t^- C^\top (C\bar{\Sigma}_t^- C^\top + \Sigma_w)^-
\end{aligned}$$

Therefore, the equations for $\bar{\Sigma}_t$ are similar to Σ_t in the Kalman filter (2.5) conlcuding that $\bar{\Sigma}_t = \Sigma_t$ for all t.

3.2 Particle-system

The mean-field process is approximated with a system of Ncontrolled interacting particles $\{X_t^1, \ldots, X_t^N\}$ according to:

$$\begin{aligned}
\text{(prediction)} \quad & (X_t^i)^- = AX_{t-1}^i + \sigma_v V_{t-1}^i \\
\textit{(update)} \quad & X_t^i = (X_t^i)^- + K_t^{(N)}(Y_t - (Y_t^i)^-), \quad (Y_t^i)^- = C(X_t^i)^- + \sigma_w W_t^i
\end{aligned}$$

for $i = 1, \ldots, N$, where V_t^i and W_t^i are independent copies of V_t and W_t, and the gain matrix is approximated using the empirical distribution of the particles:

$$K_t^{(N)} = \mathrm{Cov}(\{(X_t^i)^-, (Y_t^i)^-\}_{i=1}^N)\mathrm{Cov}(\{(Y_t^i)\}_{i=1}^N)^{-1}.$$

Thus, we end up with an interacting particle system, where in the mean-field limit as $N \to \infty$, we expect the distribution of the particles converge to the distribution of the mean-field process, hence the posterior distribution of the filter.

For numerical purposes, we define the EnKF estimate of the state of the system according to:

$$m_t^{(N)} = \frac{1}{N}\sum_{i=1}^N X_t^i$$

and the estimate for the covariance error according to:

$$\Sigma_t^{(N)} = \frac{1}{N}\sum_{i=1}^N (X_t^i - m_t^{(N)})(X_t^i - m_t^{(N)})^\top$$

We expect that, in the limit as $N \to \infty$, these estimates converge to their mean-field values $\bar{m}_t$ and $\bar{\Sigma}_t$.

The EnKF algorithm is summarized in Table 2. The following example illustrates the performance of the EnKF algorithm.

Algorithm 2 Ensemble Kalman Filter

Input: $\{Y_1, Y_2, \ldots, Y_T\}$, m_0, Σ_0, N, model parameters $(A, C, \sigma_v, \sigma_w)$
Output: particles $\{X_t^1, \ldots, X_t^N\}_{t=0}^T$
Generate samples $\{X_0^1, \ldots, X_0^N\} \sim \mathcal{N}(m_0, \Sigma_0)$
for $t = 1$ to T **do**
  Prediction step:
    $V_{t-1}^i, W_t^i \sim \mathcal{N}(0, I_n)$ for $i = 1, \ldots, N$
    $(X_t^i)^- = AX_{t-1}^i + \sigma_v V_{t-1}^i$ for $i = 1, \ldots, N$
    $(Y_t^i)^- = C(X_t^i)^- + \sigma_w \bar{W}_t^i$ for $i = 1, \ldots, N$
  Update step:
    $K_t^{(N)} = \mathrm{Cov}(\{(X_t^i)^-, (Y_t^i)^-\}_{i=1}^N)\mathrm{Cov}(\{(Y_t^i)\}_{i=1}^N)^{-1}$
    $X_t^i = (X_t^i)^- + K_t^{(N)}(Y_t - (Y_t^i)^-)$ for $i = 1, \ldots, N$
end for

3.3 Numerical example

Consider the mass-spring system introduced in 2.4. We simulate the EnKF algorithm 2 for this problem with $N = 100$ number of particles. The EnKF estimate $m_t^{(N)}$ along with the state trajectories and the observation signal, are depicted in Figure 3.1. It is observed that both KF and EnKF give an estimate that are close to each-other and the true state of the system. Next, we compare the mse in estimating the error using the KF and EnKF algorithm, by taking the average of 1000 independent simulations. The result is depicted in Figure 3.1. It is observed that both algorithms almost achieve the minimum mse $\mathrm{Tr}(\Sigma_t)$ on average.

It is also insightful to compare the empirical mean $m_t^{(N)}$ and empirical covariance $\Sigma_t^{(N)}$ evaluated on the particles with their mean-field limits $\bar{m}_t = m_t$ and $\bar{\Sigma}_t = \Sigma_t$ given by the Kalman filter.

$$\mathrm{mse}(m_t)^{(N)} := \mathbb{E}[\|\hat{m}_t^{(N)} - m_t\|^2]$$
$$\mathrm{mse}(\Sigma_t)^{(N)} := \mathbb{E}[\|\Sigma_t^{(N)} - \Sigma_t\|_F^2]$$

The result is shown in Figure 3.2. With $N = 100$ number of particles, the MSE for the mean is of the order 5×10^{-3}, and the mse for the covariance is of the order 5×10^{-4}.

As we stated earlier, it is expected that the empirical mean and the empirical covariance converge to the exact mean and covariance given by the Kalman filter as the number of particles $N \to \infty$. This is numerically illustrated in Figure 3.3.

In particular, the mes for mean and covariance is evaluated numerically for the following values for the number of particles:

$$N \in \{10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$$

The error is linear in logarithm scale with slope approximate equal to -1. This means that

$$\text{mse}(m_t)^{(N)}, \text{mse}(\Sigma_t)^{(N)} \propto \frac{1}{N}$$

as expected from Monte-Carlo approximations.

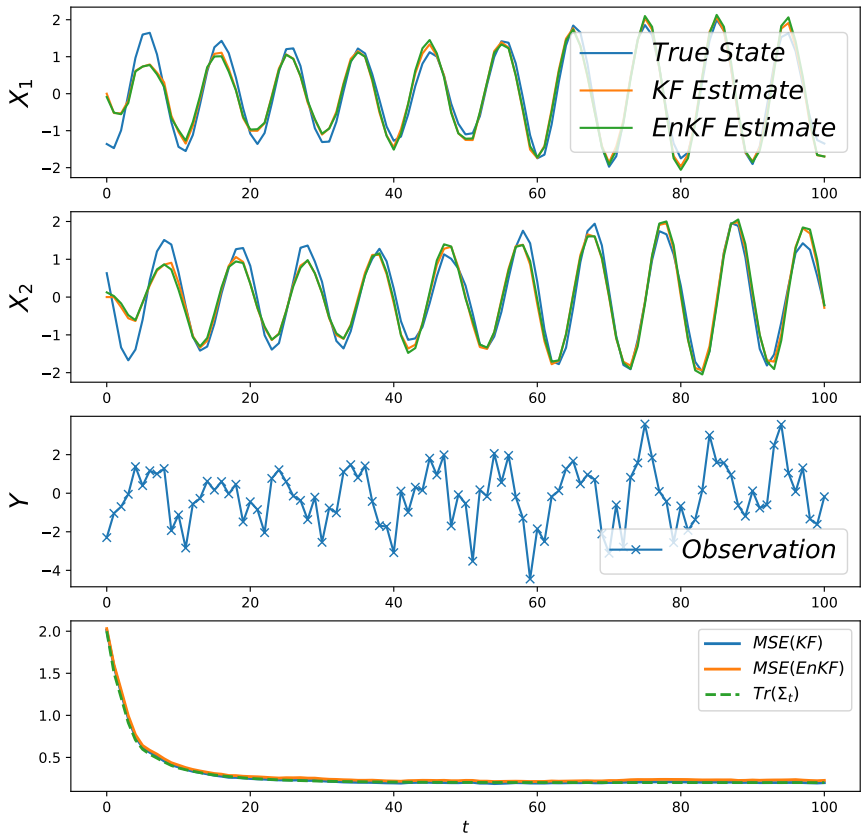

Figure 3.1: Simulation result for the EnKF algorithm for the mass-spring example 3.3. The first three graphs illustrate sample trajectory of the states of the system along with the estimates from the Kalman filter and EnKF. The last shows the average mean square error between the true state and the estimates given by the KF and EnKF algorithms. The average error is plotted against its expected value $\text{Tr}(\Sigma_t)$.

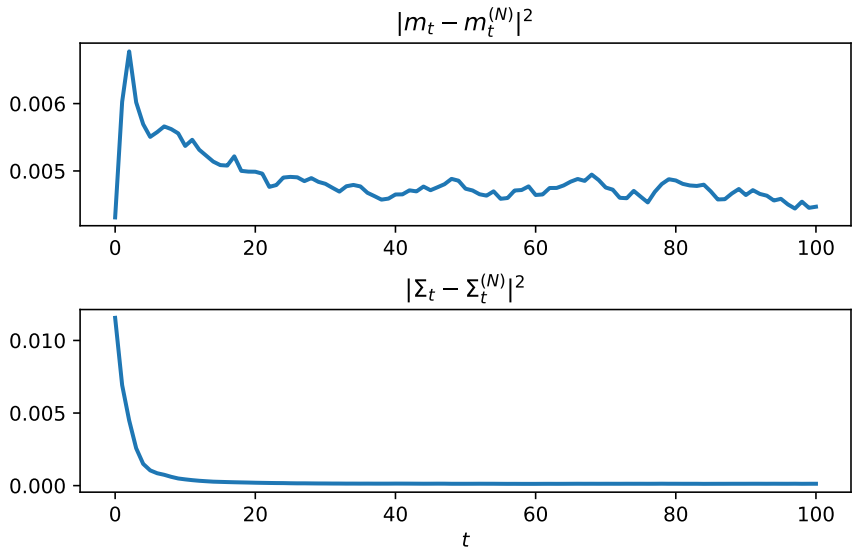

Figure 3.2: Average mean square error between the empirical mean and the empirical covariance given by the EnKF, and the mean and covariance given by the Kalman filter.

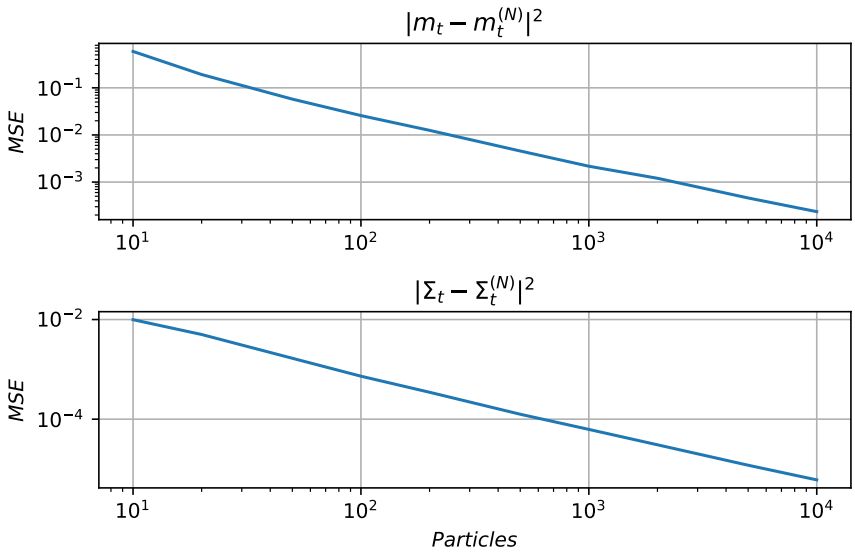

Figure 3.3: Average mean square error between the EnKF mean/covariance and KF mean and covariance, as a function of number of particles. The average is computed over 1000 independent simulations and the error is computed at $t = 100$.

Chapter 4

Optimal Transport EnKF

In this chapter, we construct and verify a variation of the EnKF algorithm that is designed based on the optimal transportation theory [26, 27] The basic idea is to create an optimal transport map that maps the prior to the posterior distribution [22, 24, 25]. This new variation was motivated by the weight degeneracy issue found in particle filters [21, 1, 3, 18]. A particle filtering algorithm proceeds by forming a weighted empirical distribution of the particles where the weights are computed using the likelihood function [8]. This is where the weight degeneracy issue can occur, meaning that one particle has much more weight than the others, causing all the other particles to be taken out during the resampling stage. In order to avoid particle degeneracy, the number of particles must increase exponentially with the number of states of the system, which is undesirable in practice.

Our aim is to use the available powerful computational hardware and software designed for machine learning tasks to get around this issue. A potential venue is brought on with the new mathematical methodology called optimal transport [26, 27]. In particular, there has been a recent introduction of variational formulations of the Bayes' formula based on optimal transportation theory [23]. We explain the procedure in the next section.

4.1 Optimal transport formulation of the conditioning

Consider the conditioning step of the mean-field process in the EnKF algorithm (3.1). We express this step, in the general form, as

$$\bar{X}_t = T_t(\bar{X}_t^-; Y_t)$$

where $T_t(x; y)$ is a (possibly stochastic) map to be designed. For example, the EnKF algorithm (3.1), the map takes the form

$$T_t(x; y) = x + K_t(y - Cx - \bar{W}_t)$$

The objective of the map is to transport the the distribution of $\bar{X}_t^-$, which is equal to $P(X_t|\mathcal{Y}_{t-1})$ to the distribution of $\bar{X}_t$, which is supposed to be equal to $P(X_t|\mathcal{Y}_t)$. We express this requirement, with the following constraint:

$$\mathbb{E}[g(\bar{X}_t; Y_t)|\mathcal{Y}_{t-1}] = \mathbb{E}[g(X_t; Y_t)|\mathcal{Y}_{t-1}], \quad \forall g$$

This constraint implies that, conditioned on $\mathcal{Y}_{t-1}$, the joint distribution of $(\bar{X}_t, Y_t)$ is equal to the joint distribution of (X_t, Y_t). Then, using $\bar{X}_t = T(\bar{X}_t^-; Y_t)$, we formulate an optimal transportation problem to find the map T that minimizes the quadratic transportation cost $\|T(\bar{X}_t^-; Y_t) - \bar{X}_t^-\|^2$, while satisfying the constraint:

$$\begin{aligned} &\min_T \mathbb{E}\left[\frac{1}{2}\|T(\bar{X}_t^-; Y_t) - \bar{X}_t^-\|^2 | \mathcal{Y}_{t-1}\right], \\ &\text{s.t} \quad \mathbb{E}\left[g(T(\bar{X}_t^-; Y_t), Y_t) - g(X_t; Y_t)|\mathcal{Y}_{t-1}\right] = 0 \quad \forall g \end{aligned}$$

We can then write this as a min-max problem

$$\min_T \max_g \mathbb{E}\left[\frac{1}{2}\|T(\bar{X}_t^-; Y_t) - \bar{X}_t^-\|^2 + g(T(\bar{X}_t^-; Y_t), Y_t) - g(X_t; Y_t)|\mathcal{Y}_{t-1}\right]$$

Assuming the duality holds, the min-max can be interchanged, concluding

$$\begin{aligned} &\max_g \min_T \mathbb{E}\left[\frac{1}{2}\|T(\bar{X}_t^-; Y_t) - \bar{X}_t^-\|^2 + g(T(\bar{X}_t^-; Y_t), Y_t) - g(X_t; Y_t)|\mathcal{Y}_{t-1}\right] \\ &= \max_g \mathbb{E}\left[\min_z \{\frac{1}{2}\|z - \bar{X}_t^-\|^2 + g(z, Y_t)\} - g(X_t; Y_t)|\mathcal{Y}_{t-1}\right] \end{aligned}$$

Define $g(x; y) = -\frac{1}{2}\|x\|^2 + f(x; y)$ to end up with the problem

$$\begin{aligned} &\max_f \mathbb{E}\left[\min_z \{-z^\top \bar{X}_t^- + f(z, Y_t)\} - f(X_t; Y_t)|\mathcal{Y}_{t-1}\right] + \text{const.} \\ &= -\min_f \mathbb{E}\left[f^*(\bar{X}_t^-, Y_t) + f(X_t; Y_t)|\mathcal{Y}_{t-1}\right] + \text{const.} \end{aligned}$$

where

$$f^*(x; y) = \max_z \{z^\top \bar{x} - f(z, y)\}$$

is the Legandre transform with respect to x. Moreover, the optimal transport map is equal to $T(x; y) = \nabla f^*(x; y)$. This is bascially an informal derivation of the well-known Brenier result [5] extended to the conditional case. A rigorous treatment is found in [7]. The optimal function f is convex. Therefore, we can change the place of f and f^* to obtain the optimization problem:

$$\begin{aligned} &\bar{X}_t = \nabla_x f(\bar{X}_t^-; Y_t) \\ &f = \arg \min_{f \in \text{CVX}_x} \mathbb{E}[f(\bar{X}_t^-, Y_t) + f^*(X_t; Y_t)|\mathcal{Y}_{t-1}] \end{aligned} \tag{4.1}$$

4.2 Restriction to quadratic functions

Now that f is specified to be convex, we can further restrict it to quadratic functions:

$$f(x,y) = \frac{1}{2}x^\top Sx + x^\top(Ky + b)$$

where S is a $n \times n$ positive symmetric matrix, K is a $n \times m$ matrix, and b is a n-dimensional vector. Such functions give rise to linear transport maps of the form

$$\nabla_x f(x,y) = Sx + Ky + b, \tag{4.2}$$

which are sufficient to represent exact transport maps from priors to posteriors whenever P_{XY} is Gaussian.

We like to find the optimal quadratic function that minimizes the optimization problem (4.1). In order to do so, we use the formula for the convex conjugate

$$\begin{aligned} f^*(x;y) &= \max_z \{z^\top \bar{x} - f(z,y)\} \\ &= \max_z \{z^\top \bar{x} - \frac{1}{2}z^\top Sz - z^\top(Ky+b)\}. \end{aligned}$$

This is a quadratic optimization problem with the optimal value

$$z = S^{-1}(x - Ky - b)$$

Plugging this back in the formula, we obtain

$$f^*(x,y) = \frac{1}{2}(x - Ky - b)^\top S^{-1}(x - Ky - b)$$

Using the quadratic form of the function $f(x;y)$ and its convex conjugate $f^*(x;y)$ the optimization problem (4.1) simplifies to:

$$\begin{aligned} \min_{S,K,b} \; & \frac{1}{2}\mathrm{Tr}(S\Sigma_x) + \frac{1}{2}\mathrm{Tr}(S^{-1}\Sigma_x) + \frac{1}{2}\mathrm{Tr}(S^{-1}K\Sigma_y K^T) \\ & - \mathrm{Tr}(S^{-1}\Sigma_{xy}K^T) + \frac{1}{2}(\tilde{b} - m_x)^T S^{-1}(\tilde{b} - m_x) \end{aligned} \tag{4.3}$$

where $m_x = \mathbb{E}[\bar{X}_t^- | \mathcal{Y}_{t-1}]$, $m_y = \mathbb{E}[\bar{Y}_t^- | \mathcal{Y}_{t-1}]$, $\tilde{b} = b - Sm_x - Km_y$, $\Sigma_x = \mathrm{Cov}(\bar{X}_t^-, \bar{X}_t^- | \mathcal{Y}_{t-1}|)$, $\Sigma_y = \mathrm{Cov}(\bar{Y}_t^-, \bar{Y}_t^- | \mathcal{Y}_{t-1}|)$ and $\Sigma_{xy} = \mathrm{Cov}(\bar{X}_t^-, \bar{Y}_t^- | \mathcal{Y}_{t-1}|)$. The optimal values of S, K, and $\tilde{b}$ are

$$\begin{aligned} S &= \Sigma_x^{-\frac{1}{2}}(\Sigma_x^{\frac{1}{2}}(\Sigma_x - \Sigma_{xy}\Sigma_y^{-1}\Sigma_{xy}^\top)\Sigma_x^{\frac{1}{2}})^{\frac{1}{2}}\Sigma_x^{-\frac{1}{2}} \\ K &= \Sigma_{xy}\Sigma_y^{-1} \\ \tilde{b} &= m_x \end{aligned}$$

Following this, we can now construct an update equation specifically for the case of quadratic functions.

$$\text{(update)} \quad \bar{X}_t = S(\bar{X}_t^- - m_x) + K(Y_t - m_y) + \tilde{b}$$

The prediction step is similar to the EnKF algorithm:

$$\text{(prediction)} \quad \bar{X}_t^- = A\bar{X}_{t-1} + \sigma_v \bar{V}_{t-1}$$

With this, the mean-field process for the optimal transport formula is derived. We can now approximate this using particles.

4.3 OT-EnKF algorithm

We are ready to describe the finite-N approximation of the mean-field process. In order to do so, we express the optimization problem (4.1) for quadratic functions where expectations are approximated in terms of particles. In particular, given samples $\{(X_t^i)^-\}_{i=1}^N$ of the mean-field process $\bar{X}_t^-$, the objective function is approximated as:

$$J^{(N)}(f) \coloneqq \frac{1}{N^2}\sum_{i,j=1}^{N} f((X_t^i)^-, (Y_t^j)^-) + \frac{1}{N}\sum_{i=1}^{N} f^*((X_t^i)^-, (Y_t^i)^-)$$

where $(Y_t^i)^- = C(X_t^i)^- + \sigma_w W_t^i$. With the quadratic form of the function and its conjugate, we have:

$$\begin{aligned} J^{(N)}(f) = \frac{1}{N^2}\sum_{i,j=1}^{N} \frac{1}{2}((X_t^i)^-)^\top S(X_t^i)^- + ((X_t^i)^-)^\top (K(Y_t^j)^- + b) \\ + \frac{1}{N}\sum_{i=1}^{N} \frac{1}{2}((X_t^i)^- - K(Y_t^i)^- - b)^\top S^{-1}((X_t^i)^- - K(Y_t^i)^- - b) \end{aligned} \tag{4.4}$$

with the update formula

$$X_t^i = S(X_t^i)^- + KY_t + b$$

It is numerically useful to center the objective function around the mean of the particles, i.e.

$$\begin{aligned} J^{(N)}(f) = \frac{1}{N^2}\sum_{i,j=1}^{N} f((X_t^i)^- - (X_t^N)^-; (Y_t^j)^- - (Y_t^N)^-) \\ + \frac{1}{N}\sum_{i=1}^{N} f^*((X_t^i)^- - (X_t^N)^-, (Y_t^i)^- - (Y_t^N)^-) \end{aligned} \tag{4.5}$$

where

$$(X_t^N)^- = \frac{1}{N}\sum_{i=1}^{N}(X_t^i)^-, \quad (Y_t^N)^- = \frac{1}{N}\sum_{i=1}^{N}(Y_t^i)^- \tag{4.6}$$

With the quadratic form of f, the first term in the objective function (4.5) simplifies to

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}((X_t^i)^- - (X_t^N)^-)^\top S((X_t^i)^- - (X_t^N)^-)$$

where the linear term cancels due to centering of X. The second term is

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(Z_t^i)^\top S^{-1} Z_t^i + \text{const.}$$

where

$$Z_t^i = (X_t^i)^- - (X_t^{(N)})^- - K((Y_t^i)^- - (Y_t^{(N)})^-) - b$$

A numerically useful way tom represent the final form of the objective is:

$$J^{(N)}(S,K,b) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\xi_t^i)^\top S\xi_t^i + \frac{1}{2}(Z_t^i)^\top S^{-1} Z_t^i \tag{4.7}$$

where

$$\begin{aligned}\xi_t^i &:= (X_t^i)^- - (X_t^N)^- \\ \eta_t^i &:= (Y_t^i)^- - (Y_t^N)^- \\ Z_t^i &= \xi_t^i - K\eta_t^i - b\end{aligned}$$

The new update equation will be

$$\text{(update)} \quad X_t^i = (X_t^N)^-) + S((X_t^i)^- - (X_t^N)^-) + K(Y_t - (Y_t^N)^-) + b$$

The prediction step is similar to the EnKF algorithm, given by:

$$\text{(prediction)} \quad (X_t^i)^- = AX_{t-1}^i + \sigma_v V_{t-1}^i$$

The overall OT-EnKF algorithm is presented in Table 3.

Algorithm 3 Optimal Transport Ensemble Kalman Filter

Input: $\{Y_1, Y_2, \ldots, Y_T\}$, m_0, Σ_0, N, model parameters $(A, C, \sigma_v, \sigma_w)$, initialization for S, K, b, learning rate, iteration number
Output: particles $\{X_t^1, \ldots, X_t^N\}_{t=0}^T$
Generate samples $\{X_0^1, \ldots, X_0^N\} \sim \mathcal{N}(m_0, \Sigma_0)$
for $t = 1$ to T **do**
  Prediction step:
    $V_{t-1}^i, W_t^i \sim \mathcal{N}(0, I_n)$ for $i = 1, \ldots, N$
    $(X_t^i)^- = AX_{t-1}^i + \sigma_v V_{t-1}^i$ for $i = 1, \ldots, N$
    $(Y_t^i)^- = C(X_t^i)^- + \sigma_w W_t^i$ for $i = 1, \ldots, N$
  Learning algorithm:
  compute $(X_t^N)^-$ and $(Y_t^N)^-$ from (4.6)
  compute $\xi_t^i = (X_t^i)^- - (X_t^N)^-$ and $\eta_t^i = (Y_t^i)^- - (Y_t^N)^-$
  for $n = 1$ to $n_{iterations}$ **do**:
    Update (S, K, b) by minimizing (4.7) using Adam
  end for
  Update step:
    $X_t^i = (X_t^N)^- + S\xi_t^i + K(Y_t - (Y_t^N)^-) + b$
end for

4.4 OT-EnKF algorithm-improved

This section proposes an improvement in implementing the OT-EnKF algorithm by using a more accurate approximation of the loss function. The idea is to use the exact observation model $Y = CX + W$ to exactly compute the expectations with respect to the Y variable. In particular, we replace $(Y^i)^-$ with $C(X^i)^- + \sigma_w W^i$ and take the expectation over W_t^i.

From the discussion of last section, the empirical approximation of the loss function is simplified to (4.7). We focus on the second term asn it is the one that involves the observation Y.

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(Z_t^i)^\top S^{-1} Z_t^i$$

where

$$\begin{aligned}\xi_t^i &:= (X_t^i)^- - (X_t^N)^- \\ \eta_t^i &:= (Y_t^i)^- - (Y_t^N)^- \\ Z_t^i &= \xi_t^i - K\eta_t^i - b\end{aligned}$$

Distributing the terms and simplifying yields four terms:

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(Z_t^i)^\top S^{-1}Z_t^i = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\xi_t^i)^\top S^{-1}\xi_t^i + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\eta_t^i)^\top K^\top S^{-1}K\eta_t^i + \frac{1}{2}b^\top S^{-1}b - \frac{1}{N}\sum_{i=1}^{N}(\xi_t^i)^\top S^{-1}K\eta_t^i$$

The first term and the third term do not involve Y. For the fourth term, substituting η_t^i with $C\xi_t^i + \sigma_w W_t^i$ and taking the expectation over W_t^i yields:

$$\frac{1}{N}\sum_{i=1}^{N}(\xi_t^i)^\top S^{-1}KC\xi_t^i$$

because expectation of W_t^i is zero. Similarly, the second term becomes

$$\frac{1}{N}\sum_{i=1}^{N}(\xi_t^i)^\top C^\top K^\top S^{-1}KC\xi_t^i + \mathrm{Tr}(K^\top S^{-1}K\Sigma_w)$$

where we used the fact that the covariance of W_t^i is identity. The third term $b^\top S^{-1}b$ is the only term in the loss function that involves b.

As a result of these simplifications, we end up with a more accurate approximation of the total loss function according to

$$\begin{aligned} J^{(N)}(S,K,b) =& \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(\xi_t^i)^\top(S + S^{-1} - S^{-1}KC - C^\top K^\top S^{-1} + C^\top K^\top S^{-1}KC)\xi_t^i \\ &+ \frac{1}{2}\mathrm{Tr}(K^\top S^{-1}K\Sigma_w) + \frac{1}{2}b^\top S^{-1}b \end{aligned} \tag{4.8}$$

The overall OT-EnKF-improved algorithm is presented in Table 4.

Algorithm 4 Optimal Transport Ensemble Kalman Filter-improved

Input: $\{Y_1, Y_2, \ldots, Y_T\}$, m_0, Σ_0, N, model parameters $(A, C, \sigma_v, \sigma_w)$, initialization for S, K, b, learning rate, iteration number
Output: particles $\{X_t^1, \ldots, X_t^N\}_{t=0}^T$
Generate samples $\{X_0^1, \ldots, X_0^N\} \sim \mathcal{N}(m_0, \Sigma_0)$
for $t = 1$ to T **do**
  Prediction step:
    $V_{t-1}^i \sim \mathcal{N}(0, I_n) \quad \text{for} \quad i = 1, \ldots, N$
    $(X_t^i)^- = AX_{t-1}^i + \sigma_v V_{t-1}^i \quad \text{for} \quad i = 1, \ldots, N$
  Learning algorithm:
  compute $(X_t^N)^-$ from (4.6)
  compute $\xi_t^i = (X_t^i)^- - (X_t^N)^-$
  for $n = 1$ to $n_{iterations}$ **do**:
    Update (S, K, b) by minimizing (4.8) using Adam
  end for
  Update step:
    $X_t^i = (X_t^N)^- + S\xi_t^i + K(Y_t - C(X_t^N)^-) + b$
end for

Chapter 5

Numerical evaluations

This chapter involves numerical experiments where the goal is to verify the OT-EnKF algorithms introduced here. The baselines are provided by the Kalman filter and EnKF.

5.1 The Learning problem

The first experiment involves solving the optimization problem(4.7) for the OT-EnKF algorithm and the the optimization problem(4.8) for the improved version. This means that we do not implement the prediction step and only implement the conditioning step for one value of the measurement. To be precise, the state random X has Gaussian distribution X $\mathcal{N}(m_0, \Sigma_0)$ with

$$m_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \Sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and the observation model is

$$Y = CX + \sigma_w W$$

with

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \sigma_w = 1$$

The algorithm proceeds by creating N samples $\{X^1, \ldots, X^N\}$ from $\mathcal{N}(m_0, \Sigma_0)$ with $N = 100$. This is enough to evaluate the loss function (4.8) to solve for (S, K, b). In order to do so, we use the PyTorch module and Adam optimization algorithm with the learning rate 0.001 for 5000 iterations. To evaluate the objective function in (4.7), we also compute $Y^i = CX^i + W^i$. Then, we follow the same procedure using the Adam algorithm to solve for (S, K, b). As a baseline, we compare the result to the exact optimal solution, when the number

of particles are infinity. The optimal parameters are:

$$\begin{aligned}\bar{S} &= \Sigma_0^{-\frac{1}{2}}(\Sigma_0^{\frac{1}{2}}\Sigma_1\Sigma_0^{\frac{1}{2}})^{\frac{1}{2}}\Sigma_x^{-\frac{1}{2}}\\ \bar{K} &= \Sigma_0 C^\top (C\Sigma_0 C^\top + \Sigma_w)^{-1}\\ \bar{b} &= m_0\end{aligned}$$

where $\Sigma_1 = \Sigma_0 - \Sigma_0 C^\top (C\Sigma_0 C^\top + \Sigma_w)^{-1} C\Sigma_0^\top$. The numerical result of optimization algorithm to find the optimal values for S, K, and b are presented in Figure 5.1,5.2, and 5.3 respectively. It is observed that the algorithms converge to a value close to their optimal values. The error is due to the fact that the number of particles N is finite.

We also demonstrate the convergence of the loss function as a function of iterations in Figure 5.4. It is observed that both techniques reach their optimal value.

In order to measure, how well we learn the optimal transport map, we compute the error

$$\frac{1}{N^2}\sum_{i,j=1}^{N} \|T(X^i;Y^j) - \bar{T}(X^i;Y^j)\|^2$$

where

$$\begin{aligned}T(x;y) &= S(x - X^{(N)}) + K(y - Y^{(N)}) + b\\ \bar{T}(x;y) &= \bar{S}(x - X^{(N)}) + \bar{K}(y - Y^{(N)}) + \bar{b}\end{aligned}$$

The numerical result, as a function of iterations is depicted in Figure 5.5. It is observed that the error in approximating the optimal transport map converges to zero.

5.2 Effect of the parameters in estimating the mean and covariance

Now that the validity of both OT-EnKF methods have been numerically verified, we like to study how different parameters affect the performance. In order to do so, we consider the same setup as the previous section. To quantify the performance, we evaluate the error approximating the optimal estimate for X, denoted by m_1, and the error in estimating the error covariance matrix Σ_1. The formula for the optimal mean and error covariance are:

$$\begin{aligned}m_1 &= m_0 + \bar{K}(Y - m_0)\\ \Sigma_1 &= \Sigma_0 - \Sigma_0 C^\top (C\Sigma_0 C^\top + \Sigma_w)^{-1} C\Sigma_0^\top\end{aligned}$$

We simulate the EnKF, OT-EnKF, and OT-EnKF-improved algorithms to approximate m_1 and Σ_1. The approximation are in terms of particles, and given

by:

$$m_1^{(N)} = \frac{1}{N}\sum_{i=1}^{N} X_1^i$$

$$\Sigma_1^{(N)} = \frac{1}{N}\sum_{i=1}^{N}(X_1^i - m_1^{(N)})(^i_1 - m_1^{(N)})^\top$$

We use $N = 100$ particles for all algorithms. The mse for approximating the mean and error covariance are defined according to

$$\mathbb{E}\|m_1^{(N)} - m_1\|^2, \quad \mathbb{E}\|\Sigma_1^{(N)} - \Sigma_1\|^2$$

The numerical result for mse in approximating the mean m_1 are depicted in Figure 5.6. The top panel shows the mse as the number of iterations grows for Σ_0 equal to identity and $\sigma_w = 1$. This is to observe what the effect of iteration number on the mse error. It is observed that, for these values of the Σ_0 and σ_w, the mse error converges to the mse value from EnKF algorithm.

Next, we repeat the experiment for a range of values for the initial covariance matrix Σ_0 and record the mse at the final iteration. The result is depicted in the second panel in Figure 5.6. And the third experiment includes changing the parameter σ_w with $\Sigma_0 = 0.1$. The result is shown in the third panel.

Same experiment, but now for mse in approximating the error covariance matrix is depicted in Figure 5.7. The results follow the same trend as figure 5.6 with respect to iterations and Σ_0. However, it diverges from it in regards to a changing σ_w.

These numerical results show an overall trend in the mse: a larger Σ_0 leads to a larger mse for all methods. This is expected as Σ_0 determines the uncertainty in the initial state. An inverse relationship is observed for σ_w. A larger σ_w leads to a lower mse. Since the estimate of the state is conditioned on the observation, the estimate will be affected by the noise. A larger observation noise will allow the learning methods to reach a lower mse for the mean, due to the added noise, which creates less strictness in finding the maps. However, a larger σ_w leads to a larger mse for the error covariance matrices for the two OT-EnKF methods, while it stays constant for the EnKF method.

5.3 Comparison with EnKF

This section examines how these methods perform on the mass-spring example of Chapter 2. We simulate the EnKF algorithm 2, the OT-EnKF algorithm 3, and the OT-EnKF-improved algorithm 4. We use $N = 100$ particles, the Pytorch module with the Adam optimization algorithm with a learning rate of 0.001, 5000 iterations, and randomly initialized S, K, b. We ran the experiments for 50 indepent simulations, and the averages of the mse's were taken. As baseline, we simulate the Kalman filter given in Table 1 and evaluate the mse approximating the Kalman mean and Kalman error covariance matrix.

The result is shown in Figure 5.8. As can be seen in the graph for mse of the mean and the error covariance matrix, the OT-EnKF works just as well as the EnKF in this example. However, the improved OT-EnKF works much better than the previous two methods. The improved OT-EnKF has a smaller mse of the means and error covariances compared against the other two methods.

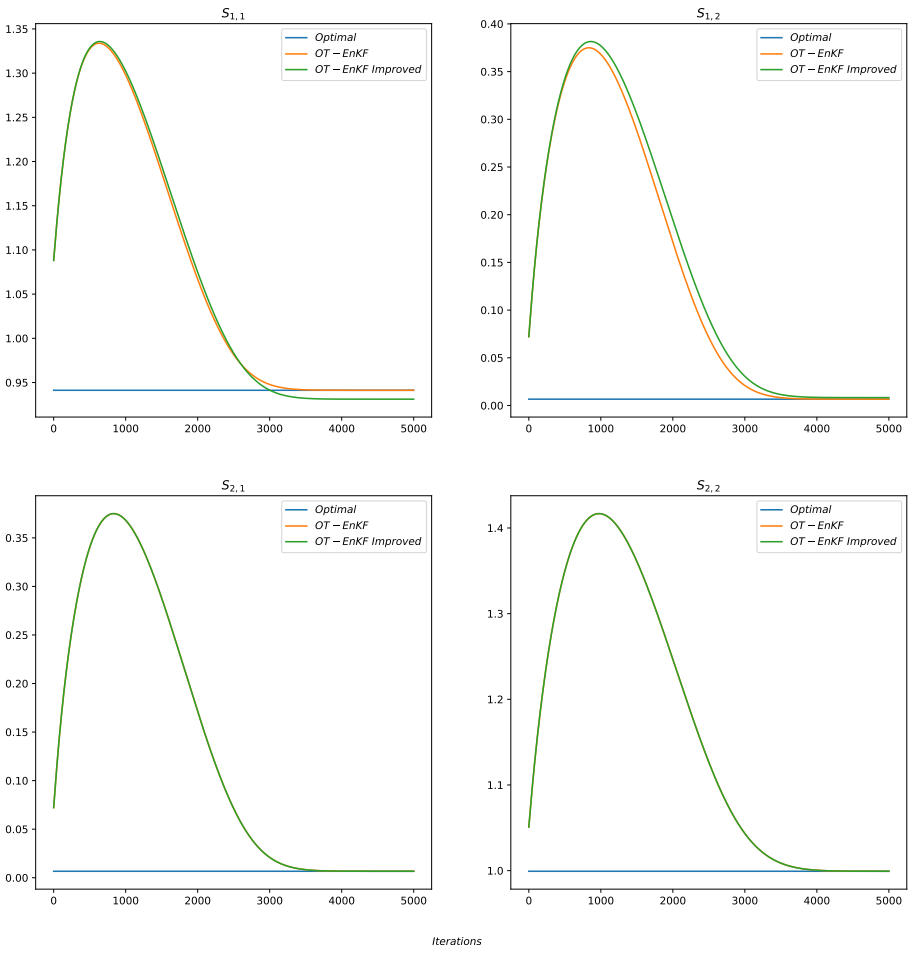

Figure 5.1: Numerical result of simulating the Adam algorithm to solve the optimization problem (4.7) and (4.8) for the OT-EnKF and OT-EnKF-improved algorithms respectively. The figure shows the convergence for learning the S parameter as a function of iterations of the ADAM algorithm.

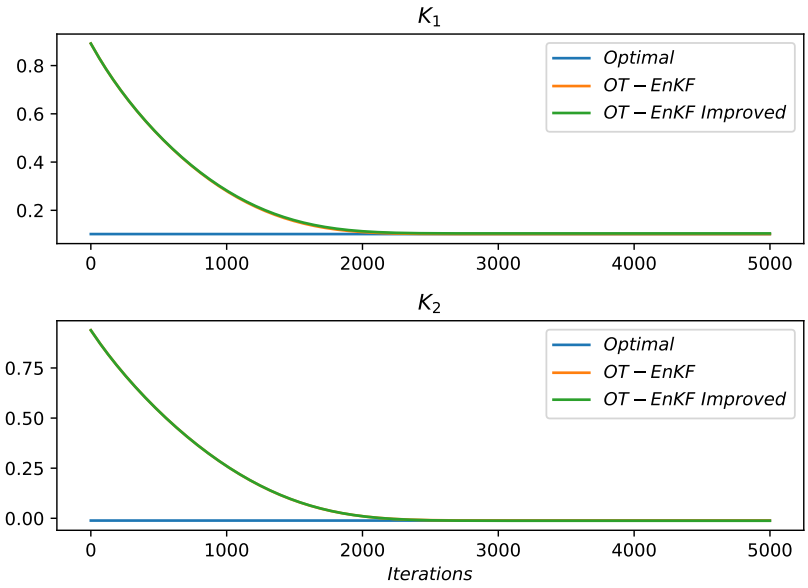

Figure 5.2: Numerical result of simulating the Adam algorithm to solve the optimization problem (4.7) and (4.8) for the OT-EnKF and OT-EnKF-improved algorithms respectively. The figure shows the convergence for learning the K parameter as a function of iterations of the ADAM algorithm.

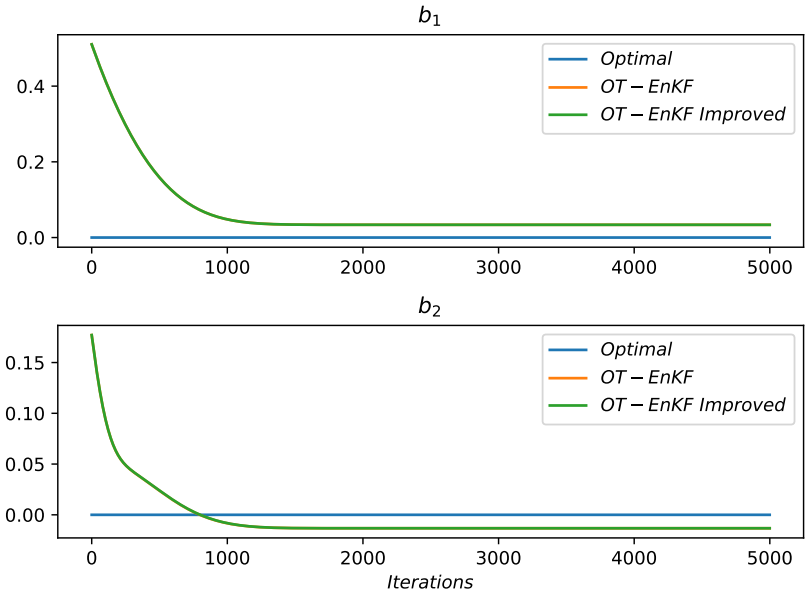

Figure 5.3: Numerical result of simulating the Adam algorithm to solve the optimization problem (4.7) and (4.8) for the OT-EnKF and OT-EnKF-improved algorithms respectively. The figure shows the convergence for learning the b parameter as a function of iterations of the ADAM algorithm.

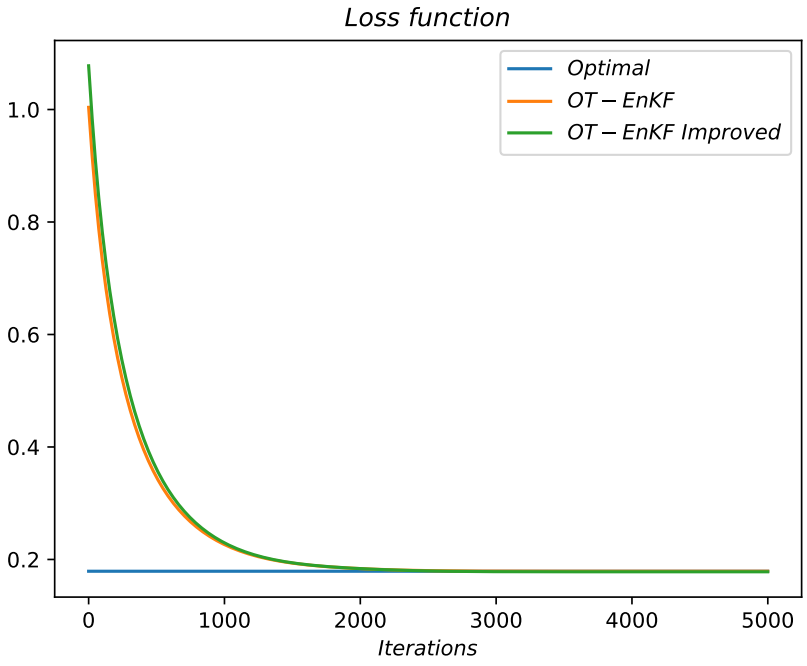

Figure 5.4: Numerical result of simulating the Adam algorithm to solve the optimization problem (4.7) and (4.8) for the OT-EnKF and OT-EnKF-improved algorithms respectively. The figure shows the value of the objective function.

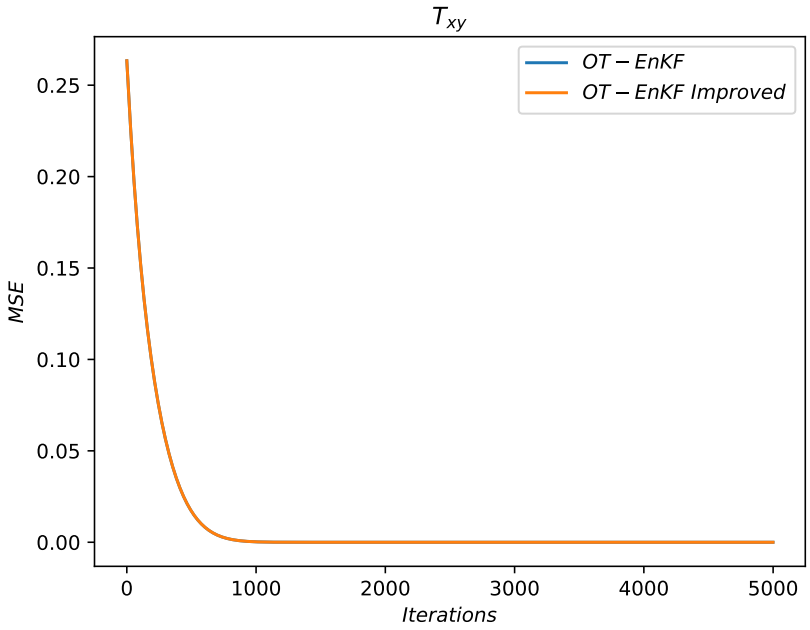

Figure 5.5: Numerical result of simulating the Adam algorithm to solve the optimization problem (4.7) and (4.8) for the OT-EnKF and OT-EnKF-improved algorithms respectively. The figure shows the error in approximating the optimal transport map.

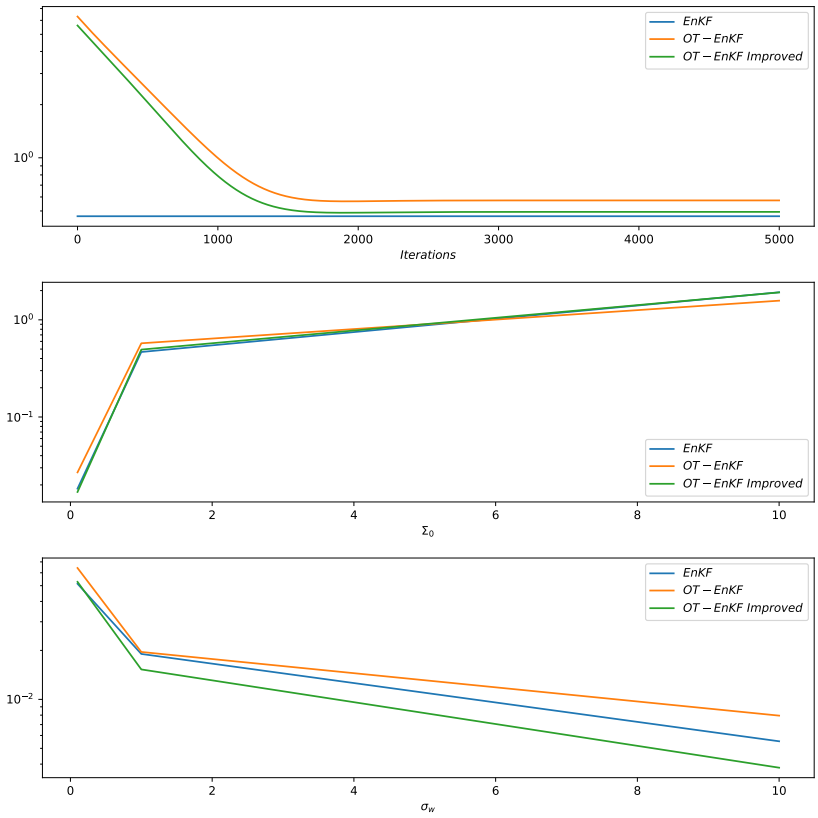

Figure 5.6: Numerical results for the mse in estimating the mean for the OT-EnKF and OT-EnKF-improved algorihtms. The baseline is provided by the EnKF algorithm. The top panel shows the mse as the number of optimization iteration grows. The second panel shows the effect of the initial covariance matrix Σ_0; and the bottom panel shows the effect of the measurement noise σ_w. The mse is computed by averaging over 50 independent simulations.

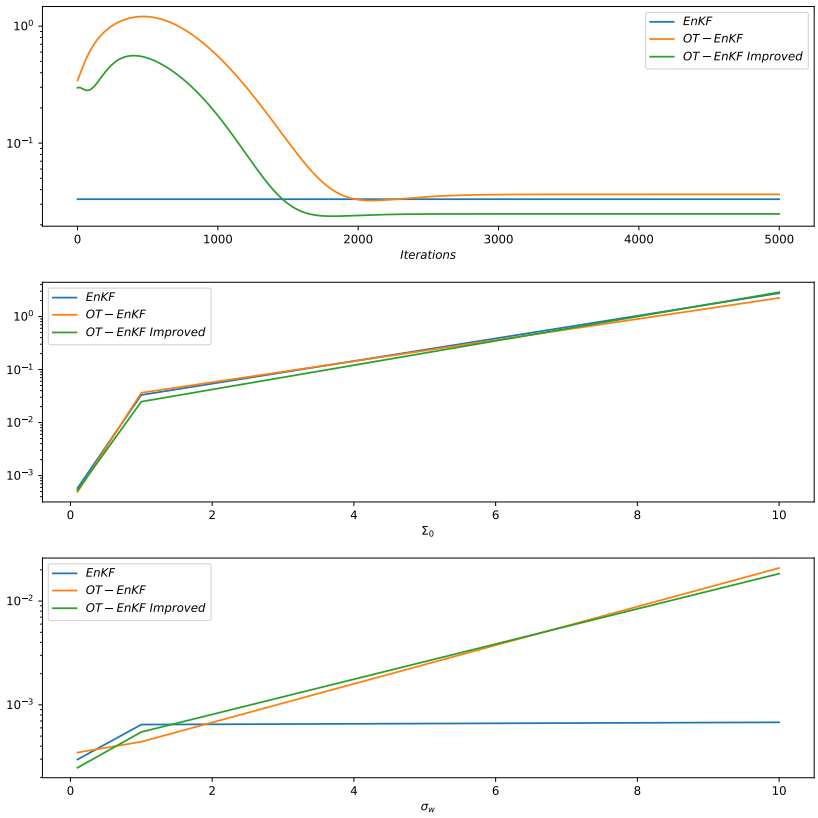

Figure 5.7: Numerical results for the mse in estimating the error covariance matrix for the OT-EnKF and OT-EnKF-improved algorihtms. The baseline is provided by the EnKF algorithm. The top panel shows the mse as the number of optimization iteration grows. The second panel shows the effect of the initial covariance matrix Σ_0; and the bottom panel shows the effect of the measurement noise σ_w. The mse is computed by averaging over 50 independent simulations.

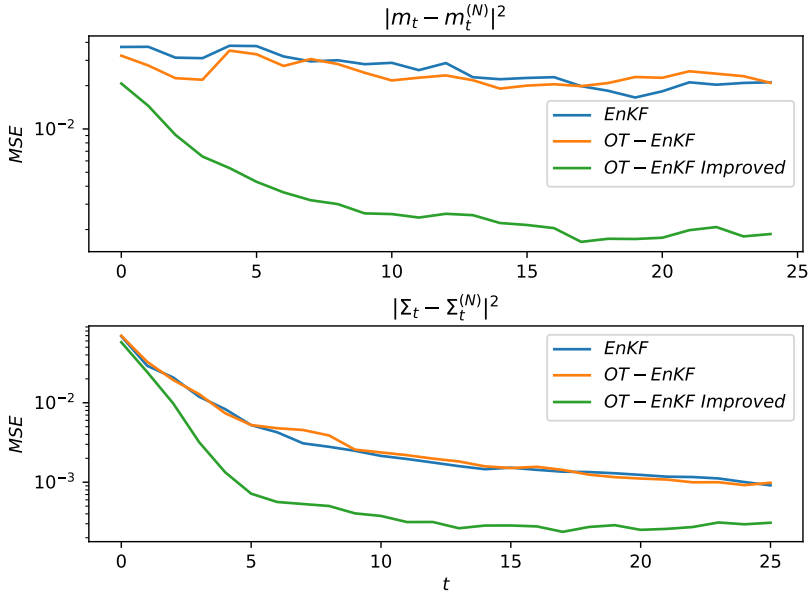

Figure 5.8: Numerical results for comparison of the EnKF, OT-EnKF, and OT-EnKF-improved algorithms. The mse for approximating the mean and error covariance matrix computed relative to the Kalman filter, are shown as a function of time. The mse is computed by averaging over 50 independent simulations.

Chapter 6

Conclusion

The development of the Kalman filter has helped predict the states at future times, allowing for better control of the dynamical system. Unfortunately, not every system is a linear system, and some are too big for the Kalman filter to calculate within a certain period of time. This motivated the work of other variations of the Kalman filter, such as the Ensemble Kalman filter that was introduced in chapter 3. We then built upon the EnKF to create a new variation, called the OT-EnKF, using the optimal transport methodology. This new technique seeks to solve an optimization problem that transports the prior distribution to the posterior distribution. The problem was striclty restriced to quadratic functions in this thesis. Two different methodologies of the OT-EnKF were created and tested for the convergence of their parameters. It was found that the parameters do indeed converge, allowing us to create a map that transports the prior distribution to the posterior distribution. It was then tested through different factors to see how the estimated state and error covariance matrix given by the Kalman filter compared against the estimated state and error covariance matrix of the two OT-EnKF techniques. The EnKF was used as a baseline. It was found that the EnKF and the OT-EnKF techniques followed similar trends for the mse of the means. The mse of the error covariance matrices were also similar with respect to a changing Σ_0. However, in regards to the mse of the error covariance matrices, we found that as the observation noise increased, the mses increased for the OT-EnKFs; however, this result stayed constant for the EnKF. It was then tested on a linear dynamical system that included Gaussian noise. Again, the mse of the state and the error covariance matrices of the Kalman filter against the two OT-EnKF techniques were taken, with the mse between the Kalman filter and EnKF again being used as a baseline. The original OT-EnKF method worked just as well as the EnKF. However, the improved OT-EnKF had a lower mse than both of the techniques.

6.1 Future Work

The scope of this thesis was to see how these OT-EnKF techniques worked when restricted to quadratic functions. Thus, a future recommendation would be to see how these techniques work when they are not restricted to these functions. Also, while we tested on linear systems, more work should be done to see how the OT-EnKF methods work against the EnKF in nonlinear systems.

We would also like to simulate a situation in which particle degeneracy would occur. We would be able to see if the two OT-EnKF methods is better than the original EnKF method in this scenario.

Bibliography

[1] T. Bengtsson, P. Bickel, and B. Li. Curse of dimensionality revisited: Collapse of the particle filter in very large scale systems. In *IMS Lecture Notes - Monograph Series in Probability and Statistics: Essays in Honor of David F. Freedman*, volume 2, pages 316–334. Institute of Mathematical Sciences, 2008.

[2] Kay Bergemann and Sebastian Reich. An ensemble kalman-bucy filter for continuous data assimilation. *Meteorologische Zeitschrift*, 21(3):213, 2012.

[3] Peter Bickel, Bo Li, and Thomas Bengtsson. Sharp failure rates for the bootstrap particle filter in high dimensions. In *Pushing the limits of contemporary statistics: Contributions in honor of Jayanta K. Ghosh*, pages 318–329. Institute of Mathematical Statistics, 2008.

[4] Adrian N Bishop and Pierre Del Moral. On the mathematical theory of ensemble (linear-gaussian) kalman-bucy filtering. *arXiv preprint arXiv:2006.08843*, 2020.

[5] Yann Brenier. Polar factorization and monotone rearrangement of vector-valued functions. *Communications on pure and applied mathematics*, 44(4):375–417, 1991.

[6] Edoardo Calvello, Sebastian Reich, and Andrew M Stuart. Ensemble kalman methods: a mean field perspective. *arXiv preprint arXiv:2209.11371*, 2022.

[7] Guillaume Carlier, Victor Chernozhukov, and Alfred Galichon. Vector quantile regression: an optimal transport approach. *The Annals of Statistics*, 44(3):1165–1192, 2016.

[8] A. M. Doucet, A.and Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.

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

[10] G. Evensen. *Data Assimilation. The Ensemble Kalman Filter*. Springer-Verlag, New York, 2006.

[11] Mohinder S Grewal and Angus P Andrews. Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[12] Bruce Hajek. *Random processes for engineers*. Cambridge university press, 2015.

[13] Peter L Houtekamer and Herschel L Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998.

[14] Peter L Houtekamer and Fuqing Zhang. Review of the ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

[15] Thomas Kailath, Ali H Sayed, and Babak Hassibi. *Linear estimation*. Number BOOK. Prentice Hall, 2000.

[16] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[17] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. 1961.

[18] Patrick Rebeschini, Ramon Van Handel, et al. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[19] S. Reich. A dynamical systems framework for intermittent data assimilation. *BIT Numerical Analysis*, 51:235–249, 2011.

[20] Sebastian Reich and Colin Cotter. *Probabilistic forecasting and Bayesian data assimilation*. Cambridge University Press, 2015.

[21] B. Ristic, S. Arulampalam, and N. Gordon. Beyond the Kalman filter. *IEEE Aerospace and Electronic Systems Magazine*, 19(7):37–38, 2004.

[22] A. Taghvaei and P. G. Mehta. An optimal transport formulation of the linear feedback particle filter. In *American Control Conference (ACC), 2016*, pages 3614–3619. IEEE, 2016.

[23] Amirhossein Taghvaei and Bamdad Hosseini. An optimal transport formulation of Bayes' law for nonlinear filtering algorithms. *arXiv preprint arXiv:2203.11869*, 2022.

[24] Amirhossein Taghvaei and Prashant G Mehta. An optimal transport formulation of the ensemble kalman filter. *IEEE Transactions on Automatic Control*, 2020.

[25] Amirhossein Taghvaei and Prashant G Mehta. Optimal transportation methods in nonlinear filtering: The feedback particle filter. *arXiv preprint arXiv:2102.10712*, 2021.

[26] Cédric Villani. *Topics in optimal transportation*. Number 58. American Mathematical Soc., 2003.

[27] Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer, 2009.

[28] J. Whitaker and T. M Hamill. Ensemble data assimilation without perturbed observations. *Monthly Weather Review*, 130(7):1913–1924, 2002.

[29] Jeffrey S Whitaker, Thomas M Hamill, Xue Wei, Yucheng Song, and Zoltan Toth. Ensemble data assimilation with the ncep global forecast system. *Monthly Weather Review*, 136(2):463–482, 2008.

Appendix A

Code Used

Listing A.1: Simulation of Dynamical System

```
def sim(A,C,Q,R,m0,P0,T):
    (m,n) = C.shape
    x = np.empty([n,T+1])
    x[:,0] = np.random.multivariate_normal(m0,P0,1)
    y = np.empty([m,T+1])
    for i in range(T):
        w = np.array([np.random.normal(0,1),np.random.normal(0,1)])
        z = np.array([np.random.normal(0,1)])
        x[:,i+1] = A@x[:,i] + Q**(1/2)@w
        y[:,i] = C@x[:,i] + R**(1/2)@z
    y[:,-1] = C@x[:,-1] + R**(1/2)@np.array([np.random.normal(0,1)])

    return x,y
```

Listing A.2: Kalman Filter

```
def kalmanFilter(A,C,P0,m0,y,Q,R,T):
    (m,n) = C.shape
    xhat = np.empty([n,T+1])
    xhat[:,0] = m0
    P = np.empty([n,n,T+1])
    P[:,:,0] = P0
    KF = np.empty([n,m,T])

    for i in range(T):
        #prediction
        xpred = A@xhat[:,i]
        Ppred = A@P[:,:,i]@A.T + Q

        #update
        K = Ppred@C.T@inv(C@Ppred@C.T+R)
        xhat[:,i+1] = xpred + K@(y[:,i+1]- C@xpred)
        P[:,:,i+1] = Ppred - K@C@Ppred
    return xhat,P
```

Listing A.3: Ensemble Kalman Filter

```
def enKF_inter(x0,Sigma0,numOfSamples,T,A,y,R,Q,C):
    #initializing arrays
    d = len(x0)
    meanX = np.zeros([d,T+1]) #making my mean array
    meanX[:,0] = x0
    #making my samples
    X = (np.random.multivariate_normal(x0,Sigma0,numOfSamples)).T
    P = np.zeros([2,2,T+1])
    P[:,:,0] = (1/(numOfSamples-1))*(X.T-meanX[:,0].T).T@(X.T-meanX[:,0].T)

    #tranposing the samples to make a n x numOfSamples array
    for i in range(T):
        #initializing my measured y, control inputs, and noises array
        W = np.random.normal(size = (len(A),numOfSamples))
        Z = np.random.normal(size = (len(C@x0),numOfSamples))

        if len(y)==1:
            Ymeasure = np.tile(y[0,i+1],(1,numOfSamples))
        else:
            Ymeasure = np.tile(y[:,i+1],(numOfSamples))

        #calculating x' and xhat'
        xprime = A@X + Q**(1/2)@W
        xhat = np.mean(xprime, axis=1, keepdims = True)


        Xhat = np.tile(xhat, numOfSamples)


        #calculating y, covariance of y and (x,y), and K
        Y = C@xprime + R**(1/2)@Z

        if len(y==1):
            yhat = np.mean(Y, keepdims = True)
            Yhat = np.tile(yhat, (1,numOfSamples))

        else:
            yhat = np.mean(Y, axis = 1, keepdims = True)
            Yhat = np.tile(yhat,numOfSamples)

        CovY = (1/(numOfSamples-1))*(Y-Yhat)@(Y-Yhat).T
        CovXY = (1/(numOfSamples-1))*(xprime-Xhat)@(Y-Yhat).T
        K = CovXY@np.linalg.inv(CovY)
```

```
        #updating my x' and getting the mean
        Xi = xprime + K@(Ymeasure-Y)
        XiMean = np.mean(Xi,axis=1)
        XiMeanTile = np.tile(XiMean[:], numOfSamples)

        #concatenating it with my meanX
        meanX[:,i+1] = XiMean[:]
        P[:,:,i+1] = (1/(numOfSamples-1))*(Xi.T-XiMean.T).T@(Xi.T-XiMean.T)

        #new x array to use in the next loop
        X = Xi
    return meanX, P
```

Listing A.4: Loss function of original OT-EnKF

```
def fstar(x_np,y_np,A,K,b,xmean,ymean):
    Asym = 0.5*(A+A.T)
    X = x_np - (K@(y_np-ymean)) - b
    return (0.5*(X.T@torch.linalg.inv(Asym)@X) + xmean.T@x_np)

def lossfun(x_np,y_np,A,K,b,numOfSamples,xmean,ymean):
    Asym = 0.5*(A+A.T)
    X = 0.5*((x_np-xmean).T@Asym@(x_np-xmean))
    Xmean = torch.trace(X)/numOfSamples
    KY = (x_np-xmean).T@(K@(y_np-ymean)+b)
    KYmean = torch.mean(KY)
    f_mean = Xmean + KYmean
    fstar_mean = torch.trace(fstar(x_np,y_np,A,K,b,xmean,ymean))
    fstar_mean = fstar_mean/(numOfSamples)
    return (f_mean + fstar_mean)
```

Listing A.5: Original OT-EnKF

```
def OT_enKF_inter(x0,Sigma0,numOfSamples,T,A,y,R,C,a,k,b,n,m):
    #initializing arrays
    d = len(x0)
    meanX = np.zeros([d,T+1]) #making my mean array
    meanX[:,0] = x0
    n_iters = 5000

    #making my samples and transposing to make 2 x numOfSamples
    X = (np.random.multivariate_normal(x0,Sigma0,numOfSamples)).T

    P = np.zeros([n,n,T+1])
    P[:,:,0] = (1/(numOfSamples-1))*(X.T-meanX[:,0].T).T@(X.T-meanX[:,0].T)

    for i in range(T):

        #initializing my measured y, control inputs, and noises array

        W = np.random.normal(size = (len(A),numOfSamples))

        Z = np.random.normal(size = (len(C@x0),numOfSamples))

        #Tiling my measured
        if len(y)==1:
            Ymeasure = np.tile(y[0,i+1],(1,numOfSamples))
        else:
            Ymeasure = np.tile(y[:,i+1],(numOfSamples))

        #calculating x' and xhat'
        xprime = A@X + Q**(1/2)@W
        xhat = np.mean(xprime, axis=1, keepdims = True)


        Xhat = np.tile(xhat, numOfSamples)

        #making xprime a float vector and then making it a torch vector
        xprimeTorch = xprime.astype(np.float32)
        xprimeTorch = torch.from_numpy(xprimeTorch)

        #making xhat a float vector and then making it a torch vector
        XhatTorch = Xhat.astype(np.float32)
        XhatTorch = torch.from_numpy(XhatTorch)

        #calculating Y and mean of Y
        Y = C@xprime + R**(1/2)@Z
```

```
if len(y==1):
    yhat = np.mean(Y, keepdims = True)
    Yhat = np.tile(yhat, (1,numOfSamples))

else:
    yhat = np.mean(Y, axis = 1, keepdims = True)
    Yhat = np.tile(yhat,numOfSamples)

#making Y and Yhat a float vector and then making it a torch vector
Y = Y.astype(np.float32)
Y_Torch = torch.from_numpy(Y)
YhatTorch = yhat.astype(np.float32)
YhatTorch = torch.from_numpy(YhatTorch)

#learning
learning_rate = 0.001
params = [a,k,b]
optimizer = torch.optim.Adam(params, lr=learning_rate)

for epoch in range(n_iters):

    #below is getting loss function
    l = lossfun(xprimeTorch,Y_Torch,a,k,b,numOfSamples,
                XhatTorch,YhatTorch)

    #gradients
    l.backward()

    #update weights
    optimizer.step()

    #zero gradients
    optimizer.zero_grad()

#updating my x' and getting the mean

knp = k.cpu().detach().numpy()
anp = a.cpu().detach().numpy()
bnp = b.cpu().detach().numpy()
Xi = anp@(xprime - xhat) + knp@(Ymeasure-yhat) + bnp

XiMean = np.mean(Xi,axis=1)

#concatenating it with my meanX
meanX[:,i+1] = XiMean[:]
```

```
        P[:,:,i+1] = (1/(numOfSamples-1))*(Xi.T-XiMean.T).T@(Xi.T-XiMean.T)

        #new x array to use in the next loop
        X = Xi

    return meanX, P
```

Listing A.6: Loss function of improved OT-EnKF

```
def imp_lossfun(x_np,y_np,A,K,b,numOfSamples,xmean,ymean,C,R):
    Asym = 0.5*(A+A.T)

    I = 0.5*((x_np-xmean).T@Asym@(x_np-xmean))
    I_mean = torch.trace(I)/numOfSamples

    II_1 = 0.5*((x_np-xmean).T@torch.linalg.inv(Asym)@(x_np-xmean))
    II_1_mean = torch.trace(II_1)/numOfSamples

    II_2 = -(x_np-xmean).T@torch.linalg.inv(Asym)@K@C@(x_np-xmean)
    II_2_mean = torch.trace(II_2)/numOfSamples

    II_31 = 0.5*(x_np-xmean).T@C.T@K.T@torch.linalg.inv(Asym)@K@C@(x_np-xmean)
    II_32 = 0.5*torch.trace(K.T@torch.linalg.inv(Asym)@K@R)
    II_3_mean = torch.trace(II_31)/numOfSamples + II_32

    II_4 = 0.5*(xmean-b).T@torch.linalg.inv(Asym)@(xmean-b)
    II_4_mean = torch.trace(II_4)/numOfSamples

    f_mean = I_mean + II_1_mean + II_2_mean + II_3_mean +II_4_mean
    return (f_mean)
```

Listing A.7: Improved OT-EnKF

```
def impOT_enKF_inter(x0,Sigma0,numOfSamples,T,A,y,R,C,a,k,b,n,m):
    #initializing arrays
    d = len(x0)
    meanX = np.zeros([d,T+1]) #making my mean array
    meanX[:,0] = x0
    n_iters = 5000

    CTorch = C.astype(np.float32)
    CTorch = torch.from_numpy(CTorch)
    RTorch = R.astype(np.float32)
    RTorch = torch.from_numpy(RTorch)

    #making my samples and transposing to make 2 x numOfSamples
    X = (np.random.multivariate_normal(x0,Sigma0,numOfSamples)).T

    P = np.zeros([n,n,T+1])
    P[:,:,0] = (1/(numOfSamples-1))*(X.T-meanX[:,0].T).T@(X.T-meanX[:,0].T)

    for i in range(T):

        #initializing my measured y, control inputs, and noises array

        W = np.random.normal(size = (len(A),numOfSamples))

        Z = np.random.normal(size = (len(C@x0),numOfSamples))

        #Tiling my measured
        if len(y)==1:
            Ymeasure = np.tile(y[0,i+1],(1,numOfSamples))
        else:
            Ymeasure = np.tile(y[:,i+1],(numOfSamples))

        #calculating x' and xhat'
        xprime = A@X + Q**(1/2)@W
        xhat = np.mean(xprime, axis=1, keepdims = True)


        Xhat = np.tile(xhat, numOfSamples)

        #making xprime a float vector and then making it a torch vector
        xprimeTorch = xprime.astype(np.float32)
        xprimeTorch = torch.from_numpy(xprimeTorch)

        #making xhat a float vector and then making it a torch vector
```

```
XhatTorch = Xhat.astype(np.float32)
XhatTorch = torch.from_numpy(XhatTorch)

#calculating Y and mean of Y
Y = C@xprime + R**(1/2)@Z

if len(y==1):
    yhat = np.mean(Y, keepdims = True)
    Yhat = np.tile(yhat, (1,numOfSamples))

else:
    yhat = np.mean(Y, axis = 1, keepdims = True)
    Yhat = np.tile(yhat,numOfSamples)

#making Y and Yhat a float vector and then making it a torch vector
Y = Y.astype(np.float32)
Y_Torch = torch.from_numpy(Y)
YhatTorch = yhat.astype(np.float32)
YhatTorch = torch.from_numpy(YhatTorch)

#learning
learning_rate = 0.001
params = [a,k,b]
optimizer = torch.optim.Adam(params, lr=learning_rate)

for epoch in range(n_iters):

    #below is getting loss function
    l = imp_lossfun(xprimeTorch,Y_Torch,a,k,b,numOfSamples,
                    XhatTorch,YhatTorch,CTorch,RTorch)
    #gradients
    l.backward()

    #update weights
    optimizer.step()

    #zero gradients
    optimizer.zero_grad()

#updating my x' and getting the mean

knp = k.cpu().detach().numpy()
anp = a.cpu().detach().numpy()
bnp = b.cpu().detach().numpy()
Xi = anp@(xprime - xhat) + knp@(Ymeasure-C@xhat) + bnp
```

```
        XiMean = np.mean(Xi, axis=1)

        #concatenating it with my meanX
        meanX[:, i+1] = XiMean[:]
        P[:,:, i+1] = (1/(numOfSamples-1))*(Xi.T-XiMean.T).T@(Xi.T-XiMean.T)

        #new x array to use in the next loop
        X = Xi

    return meanX, P
```