M2R Second Year Group Project

Imperial College London

Department of Mathematics

Bayesian Filtering

Author:
Vignesh Balaji(01849526), Tony Dam(01852595), Varsha Otta(01857743), Xiandong Zou(01873814), Tian Zhu(01871483), Tristan Peroy(01854740)

Date: June 27, 2023

Contents

1 Introduction

In statistics, we often need to estimate an unknown probability density function (PDF) recursively over time from incoming measurements and some mathematical process models.
Bayesian filtering is a general probabilistic approach under The Makarov hypothesis for estimating an PDF recursively from noisy measurements, it is generally based on the theorem of Bayesian statistics, the numerical analysis and some results from stochastic processes.

In this report we will take an overview to filtering problem, Bayesian Filtering structure and introduce several filtering Algorithms by equation derivation and coding simulation for these problems:

- The Kalman filter (KF): gives a closed form solution to the linear filtering problem.
- The extended Kalman filter (EKF): results in a approximation to a non-linear and non- filtering problem by linearization.
- Particle filters (particle filter): results in Monte Carlo approximations based on different sampling methods for a problem in which the approximations are not appropriate.
- The unscented Kalman filter (UKF): results in a approximation to the propagation of densities from non-linearities of measurement by unscented transform. (brief description only)

We will consider the applications to Car Tracking Model for linear systems, and Noisy Pendulum Model for non-linear systems to explain and simulate those filtering Algorithms.

1.1 Targets

- Understand Filtering problem and the basic structure of Bayesian Filtering.
- Learn the origin of KF, EKF, PF and their derivation step.
- Use Car Tracking Model and Noisy Pendulum Model as example to understand the application of filters in real world problem.
- Compare the advantages and disadvantages of several filters through code simulation.

2 Overview of Filtering

2.1 What is Filtering

Optimal filtering problem (or filtering problem) are considered as statistical inversion problems.[1] The problem is estimating the hidden state of the time-varying system $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \mathbf{x}_1, \dots \mathbf{x}_T\}$ from the noisy observed measurement $\mathbf{y}_{0:T} = \{\mathbf{y}_0, \mathbf{y}_1, \dots \mathbf{y}_T\}$.

2.2 Structure

The basic structure of Bayesian Filtering dynamic model can be described as a **Hidden Markov Model** (HMM),[3] the term *hidden* stand for the unknown true state $\mathbf{x}_k$. The HMM model has two important properties:

- **The Markov Property (Markovianity)**: the true state depend on the previous state only, can be written as[4]:

$$p\left(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1}\right) = p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}\right) \tag{1}$$

- **Conditional Independence of Measurements**: the measurement depend on the current state only, can be written as:

$$p\left(\mathbf{y}_t \mid \mathbf{x}_{1:t}, \mathbf{y}_{1:t-1}\right) = p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) \tag{2}$$

In the Bayesian filtering algorithm a two-step cycle repeatedly applied:

1) the **predict step**, where we make an estimate about the current state, with given only the prior measurements.

2) the **update step**, where we use the current measurement to estimate the current state.

Through such estimation cycle we can continuously estimate the future state and optimize the estimation model.

2.3 Filtering model

Bayesian filtering is the Bayesian way to form optimal filtering. Apply the Bayes' rule to the problem, we can have the joint posterior distribution of all the states[1]:

$$p\left(\mathbf{x}_{0:T} \mid \mathbf{y}_{1:T}\right) = \frac{p\left(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}\right) p\left(\mathbf{x}_{0:T}\right)}{p\left(\mathbf{y}_{1:T}\right)} \tag{3}$$

where $p\left(\mathbf{x}_{0:T}\right)$ is the prior distribution defined by the dynamic model
$p\left(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}\right)$ is the likelihood model for the measurements
$p\left(\mathbf{y}_{1:T}\right)$is the normalization constant: $p\left(\mathbf{y}_{1:T}\right) = \int p\left(\mathbf{y}_{1:T} \mid \mathbf{x}_{0:T}\right) p\left(\mathbf{x}_{0:T}\right) d\mathbf{x}_{0:T}$

The states and measurements model considering will be assumed into these:

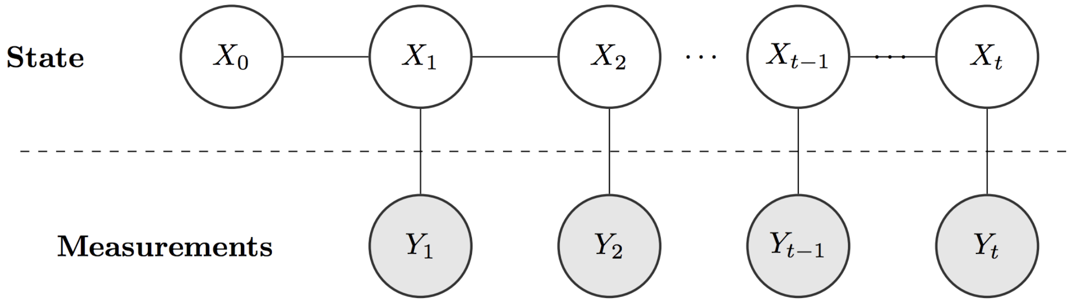


Figure 1: Graph representation of Bayesian filter structure from John Lambert `https://johnwlambert.github.io/bayes-filter/#state-estimation`

- **An initial distribution** state the *prior probability distribution* $p(\mathbf{x}_0)$ of the hidden state $\mathbf{x}$ at the start.
- **A dynamic model** state the dynamics of the system as the transition probability distribution $p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$
- **A measurement model** state the dependence of the measurement $\mathbf{y}_k$ on the corresponding state $\mathbf{x}_k$ as the conditional probability distribution $p(\mathbf{y}_k \mid \mathbf{x}_k)$

Thus we will be working on this general probabilistic state space model as following:

$$\mathbf{x}_0 \sim p(\mathbf{x}_0) \tag{4}$$
$$\mathbf{x}_k \sim p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \tag{5}$$
$$\mathbf{y}_k \sim p(\mathbf{y}_k \mid \mathbf{x}_k) \tag{6}$$

For **Filtering distributions** we are aiming to compute the marginal distributions of the current state $\mathbf{x}_k$ given the corresponding current and previous measurement $\mathbf{y}_{0:k} = \{\mathbf{y}_0, \mathbf{y}_1, \dots \mathbf{y}_k\}$. It will be written as:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}), \qquad k = 1, \dots, T \tag{7}$$

For **Prediction distributions** we are aiming to compute the marginal distributions of the future state $\mathbf{x}_{k+n}$, n steps after the current measurement. It will be written as:

$$p(\mathbf{x}_{k+n} \mid \mathbf{y}_{1:k}), \qquad k = 1, \dots, T \qquad n = 1, 2, \dots \tag{8}$$

For **Smoothing distributions** we are aiming to compute the marginal distributions of the previous state $\mathbf{x}_k$, with the given measurement in a larger interval $\mathbf{y}_{1:T} = \{\mathbf{y}_0, \mathbf{y}_1, \dots \mathbf{y}_T\}$ for some $T > k$. It will be written as:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:T}), \qquad k = 1, \dots, T \tag{9}$$

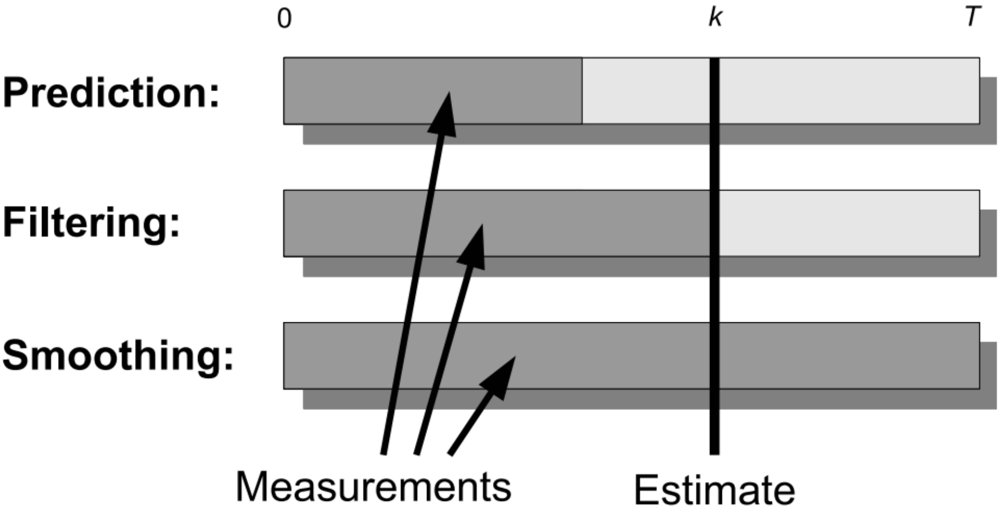


Figure 2: Graph representation of the difference for filtering, prediction and smoothing from Simo Sarkka Bayesian filtering and smoothing. page 11

2.4 Derivation of Predict Step

Start with the previous timestep posterior, by marginalization we have[3]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int_{x_{t-1}} p(\mathbf{x}_t, \mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{10}$$

Using the chain rule we have:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int_{x_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{y}_{1:t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{11}$$

According to **Markovianity** we can simplify to get the **Predict Step**:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int_{x_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \tag{12}$$

2.5 Derivation of Update Step

In the update step we will express the posterior by Bayes' rule, start with marginalization we have[3]:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_{1:t} \mid \mathbf{x}_t)\, p(\mathbf{x}_t)}{p(\mathbf{y}_{1:t})} = \frac{p(\mathbf{y}_{1:t} \mid \mathbf{x}_t)\, p(\mathbf{x}_t)}{\int_{x_t} p(\mathbf{y}_{1:t} \mid \mathbf{x}_t)\, p(\mathbf{x}_t)\, d\mathbf{x}_t} \tag{13}$$

Using the chain rule we have:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1})\, p(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t)\, p(\mathbf{x}_t)}{\int_{x_t} p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1})\, p(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t)\, p(\mathbf{x}_t)\, d\mathbf{x}_t} \tag{14}$$

According to **Conditional Independence of Measurements** we have $p\left(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{1:t-1}\right) = p\left(\mathbf{y}_t \mid \mathbf{x}_t\right)$, so we can get:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right)}{\int_{x_t} p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) d\mathbf{x}_t} \tag{15}$$

By Bayes' rule we also have:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) = \frac{p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right)}{\int_{x_t} p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) d\mathbf{x}_t} \tag{16}$$

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) \int_{x_t} p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) d\mathbf{x}_t = p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) \tag{17}$$

By marginalization we have $\int_{x_t} p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) d\mathbf{x}_t = p\left(\mathbf{y}_{1:t-1}\right)$, so we can simplify to get:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) p\left(\mathbf{y}_{1:t-1}\right) = p\left(\mathbf{y}_{1:t-1} \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t\right) \tag{18}$$

Substitute into (15) we have:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) p\left(\mathbf{y}_{1:t-1}\right)}{\int_{x_t} p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) p\left(\mathbf{y}_{1:t-1}\right) d\mathbf{x}_t} \tag{19}$$

Since $p\left(\mathbf{y}_{1:t-1}\right)$ is independent upon x, so we can cancel this term to get the simplify **Update step**:

$$p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t}\right) = \frac{p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right)}{\int_{x_t} p\left(\mathbf{y}_t \mid \mathbf{x}_t\right) p\left(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}\right) d\mathbf{x}_t} \tag{20}$$

2.6 Summary

In practice, the filtering distribution often cannot be computed due to different parameters or the complexity of the computation. However, under certain conditions(Linear or Gaussian), by several algorithms we can simplify the computation to get the filtering distribution.

In the following sections, for different models, we will introduce KF, Particle Filter and KF's extensions EKF respectively, and corresponding application examples with coding simulation.

3 Kalman Filter

The Kalman filter is an algorithm used to provide an estimate $\hat{x_n}$ to the true state of a system x_n which cannot be directly observed[5]. It combines models of the system and noisy measurements of linear functions of parameters, and consists of two steps; a **predict** and an **update** as it is a Bayesian Filter, which when applied recursively produces the desired estimate. This filter has numerous applications in areas such as finance, navigation and signal processing[6].

Assume we have a linear Gaussian state space model[1]:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k \end{aligned} \tag{21}$$

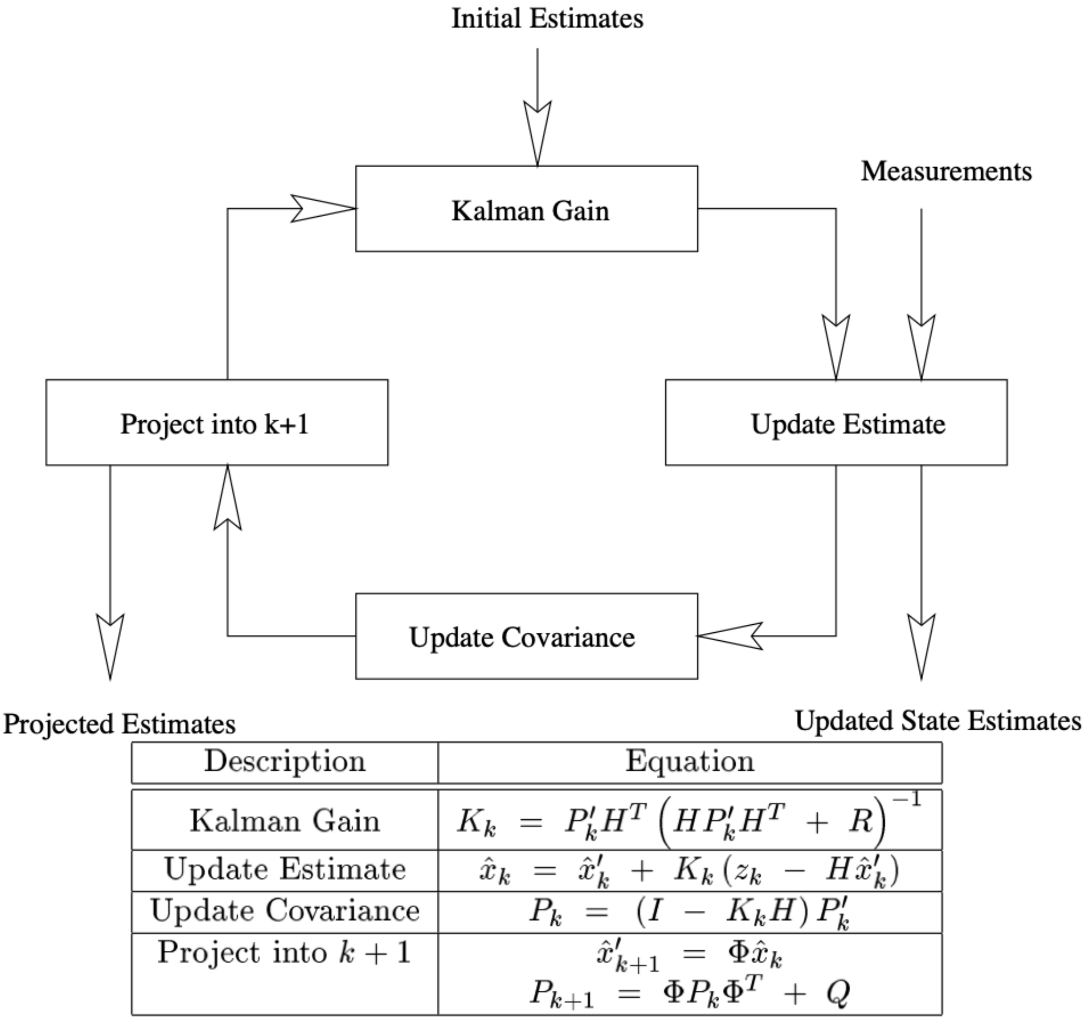
- $\mathbf{x}_k \in \mathbb{R}^n$ is the state
- $\mathbf{A}_{k-1}$ is the transition matrix of the dynamic model
- $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the process noise
- $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement
- $\mathbf{H}_k$ is the measurement model matrix
- $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise

Then the figure on the next page illustrates the outline of the algorithm.

In the diagram there is notation which we don't follow in this report. P_k is the error covariance matrix at time k, z_k is the measurement at time k, Φ is the transition matrix of the dynamic model and both **Q** and **R** are the covariance matrices for the process noise and the measurement noise respectively, they should ideally be sub scripted as seen later on.

Outline:

1. Predict the state at time step k given all the measurements up to time step k-1, i.e. $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$
2. Using the predicted state from step 1 calculate the predicted likelihood or the measurement at time step k given all the measurements beforehand, i.e. $p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})$
3. Finally we update the predicted state given the observed measurement at time step k, i.e. $p(\mathbf{x}_k \mid \mathbf{y}_{1:k})$



Description	Equation
Kalman Gain	$K_k = P'_k H^T \left(H P'_k H^T + R\right)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}'_k + K_k (z_k - H\hat{x}'_k)$
Update Covariance	$P_k = (I - K_k H) P'_k$
Project into $k+1$	$\hat{x}'_{k+1} = \Phi\hat{x}_k$ $P_{k+1} = \Phi P_k \Phi^T + Q$

Figure 3: The Kalman Process from `https://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf`

3.1 The Kalman Algorithm

Proposition 3.1 (Kalman Filter)[1] *The Bayesian filtering equations for the linear filtering model (21) can be evaluated in closed form and the resulting distributions are Gaussian*:

$$
\begin{aligned}
p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k^-, \mathbf{P}_k^-), \\
p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1}) &= \mathrm{N}(\mathbf{y}_k \mid \mathbf{H}_k \mathbf{m}_k^-, \mathbf{S}_k) \\
p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) &= \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k),
\end{aligned}
\tag{22}
$$

Computed using the following ***prediction*** *step, given a prior distribution* $\mathbf{x}_0 \sim N(\mathbf{m}_0, \mathbf{P}_0)$*:*

$$\mathbf{m}_k^- = \mathbf{A}_{k-1}\mathbf{m}_{k-1}, \mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1},$$

And the following **update** *step*:

$$\begin{aligned}\mathbf{v}_k &= \mathbf{y}_k - \mathbf{H}_k\mathbf{m}_k^-,\\ \mathbf{S}_k &= \mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k,\\ \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}_k^T\mathbf{S}_k^{-1},\\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k,\\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T,\end{aligned} \tag{23}$$

Aim: Given that the prior distribution is Gaussian we get that passing this through the state model will result in a Gaussian distribution for $\boldsymbol{x_k}$ as we have a **linear combination** of **Gaussian distributions**. Hence the aim of the derivation is to calculate the mean and covariance of each of the distributions.

In order to calculate the mean and covariance during the update step we will require the following lemma.

Lemma 3.2 (Gaussian Conditioning Formula)[6] *Suppose we have the following multivariate normal random variable*:

$$\mathbf{X} = \begin{pmatrix}\mathbf{X_1}\\ \mathbf{X}_2\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}, \begin{pmatrix}\sum_{11} & \sum_{12}\\ \sum_{21} & \sum_{22}\end{pmatrix}\right) \tag{24}$$

Then the random variable $\hat{\mathbf{X}}$ *follows a Gaussian distribution*:

$$\hat{\boldsymbol{X}} = (\boldsymbol{X}_1 \mid \boldsymbol{X}_2 = \boldsymbol{Y}) \sim \mathrm{N}\left(\hat{\mu}, \hat{\Sigma}\right) \tag{25}$$

where $\boldsymbol{Y}$ *is a random variable and*:

$$\hat{\mu} = \mu_1 + \Sigma_{11}\Sigma_{22}^{-1}(y - \mu_2) \tag{26}$$

$$\hat{\Sigma} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} \tag{27}$$

Derivation of Proposition 3.1

Prediction Step:

We follow the aim and calculate the expectation and covariance of the future state given the previous measurements.

By linearity of expectation and also given that the process noise has zero mean we obtain the following expression for the expectation of $p(\mathbf{x}_k \mid \mathbf{y}_{1:k-1})$:

$E(\boldsymbol{X}_k \mid \boldsymbol{Y}_{1:k-1}) = \boldsymbol{A}_{k-1}\boldsymbol{m}_{k-1}$

Using properties of covariance and exploiting the fact that the process noise has unit covariance we obtain the following expression:

$Cov\,(\boldsymbol{X}_n \mid \boldsymbol{Y}_{1:k-1}) = \boldsymbol{A}_{k-1}\boldsymbol{P}_{k-1}\boldsymbol{A}_{k-1}^T + \boldsymbol{Q}_{k-1}$

These two expressions make up $\boldsymbol{m}_k^-$ and $\boldsymbol{P}_k^-$ respectively, from this we can calculate the mean and covariance for $p\,(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})$. We pass the expressions from above into the measurement model and then apply the properties of expectation to obtain:

$E\,(\boldsymbol{Y}_k \mid \boldsymbol{Y}_{1:k-1}) = \boldsymbol{H}_k\boldsymbol{A}_{k-1}\boldsymbol{m}_{k-1}$

$Cov\,(\boldsymbol{X}_n \mid \boldsymbol{Y}_{1:k-1}) = \boldsymbol{H}_k\boldsymbol{P}_{k-1}^-\boldsymbol{H}_k^T + \boldsymbol{R}_k$

Where we use $\boldsymbol{P}_{k-1}^-$ for convenience, so that the algebra isn't as complicated. The covariance expression gives us $\boldsymbol{S}_k$ which is used in the update step.

Update Step:

In this step we must use the Gaussian condition formula or **Lemma 3.2**. In order to see how it may be useful we can rewrite $p\,(\mathbf{x}_k \mid \mathbf{y}_{1:k})$ as:

$$p(\mathbf{x}_k \mid \mathbf{y}_k, \mathbf{y}_{1:k-1}) = \frac{p(\mathbf{x}_k, \mathbf{y}_k \mid \mathbf{y}_{1:k-1})}{p(\mathbf{y}_k \mid \mathbf{y}_{1:k-1})} \tag{28}$$

using Bayes' rule. Therefore we can apply the Gaussian Conditioning formula, where $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are as follows:

$$\begin{pmatrix} \mathbf{X_1} \\ \mathbf{X}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{X_k} \mid \mathbf{Y_{1:k-1}} \\ \mathbf{Y_k} \mid \mathbf{Y_{1:k-1}} \end{pmatrix} \tag{29}$$

The cross covariance terms Σ_{12} and Σ_{21} are $\boldsymbol{P}_k^-\boldsymbol{H}_k^T$ and $\boldsymbol{H}_k\boldsymbol{P}_k^{-T}$ respectively[6]. Then plugging in the covariance of $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ into the equation we end up with the desired expressions for $\boldsymbol{m}_k^-$ and $\boldsymbol{P}_k^-$ where $\boldsymbol{K}_k$ or the Kalman gain and $\boldsymbol{v}_k$ are embedded in these equations.

3.2 Summary

The Kalman filter is an efficient recursive algorithm which minimizes the mean squared error of the state vector[11]. The filter is very powerful in the sense that it supports estimations of past, present and even future states[11]. However, the version presented in this section relies on following a linear Gaussian model, in the next section we will discuss how it can also be used in the non-linear case.

4 Extended Kalman Filter

Suppose we have a state space model where the dynamic and measurement models are non-linear. This tends to be the case in various applications, as systems may involve factors such as angle. We want an estimate to the true state of the system, yet the Kalman Filter cannot be applied because the non-linearly leads to a non-Gaussian posterior distribution. We can, however, apply Taylor series approximations using an EKF algorithm, to approximate the filtering distributions as Gaussian distributions[1]:

$$p(\mathbf{x}_k \mid \mathbf{y}_{1:k}) \simeq \mathrm{N}(\mathbf{x}_k \mid \mathbf{m}_k, \mathbf{P}_k) \tag{30}$$

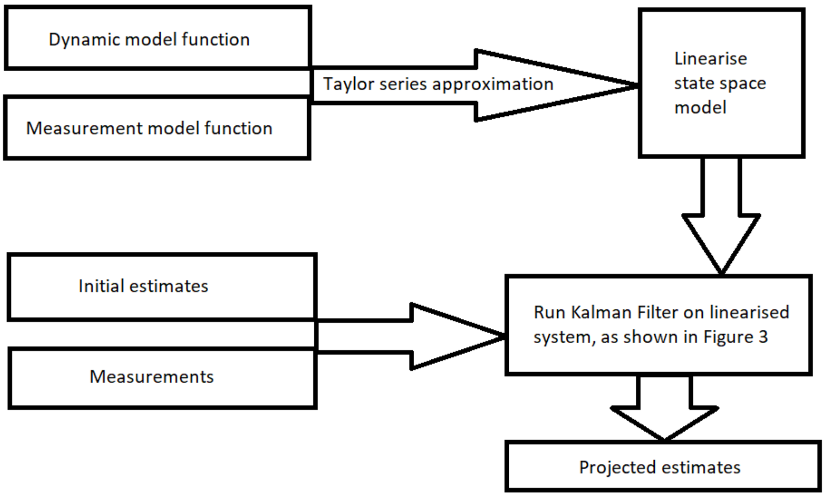

Dynamic model function
Measurement model function
Taylor series approximation
Linearise state space model
Initial estimates
Measurements
Run Kalman Filter on linearised system, as shown in Figure 3
Projected estimates

Figure 4: The Extended Kalman Process

4.1 EKF Algorithms

We will consider systems with additive and non-additive noise separately. For additive noise, we have[1]:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \tag{31}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k \tag{32}$$

And for non-additive noise, we have[1]:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}), \tag{33}$$
$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) \tag{34}$$

- $\mathbf{f}(\cdot)$ is the dynamic model function
- $\mathbf{h}(\cdot)$ is the measurement model function

Note that the functions $\mathbf{f}$ and $\mathbf{h}$ are in fact $\mathbf{f}_k(\cdot)$ and $\mathbf{h}_k(\cdot)$ as they can depend on k; denoted as $\mathbf{f}$ and $\mathbf{h}$ respectively for simplicity of notation.

Algorithm 1 First order EKF filter for systems with additive noise[1]

```
Prediction:
    $\mathbf{m}_k^- = \mathbf{f}(\mathbf{m}_{k-1}),$
    $\mathbf{P}_k^- = \mathbf{F}_\mathbf{x}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{F}_\mathbf{x}^\top(\mathbf{m}_{k-1}) + \mathbf{Q}_{k-1} \qquad (35)$
Update:
    $\mathbf{v}_k = \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-),$
    $\mathbf{S}_k = \mathbf{H}_\mathbf{x}(\mathbf{m}_k^-)\mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{m}_k^-) + \mathbf{R}_k,$
    $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{m}_k^-)\mathbf{S}_k^{-1},$
    $\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k,$
    $\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top \qquad (36)$
    where the Jacobian $\mathbf{F}_\mathbf{x}$ of $\mathbf{f}(\mathbf{x})$ is evaluated at $\mathbf{x} = \mathbf{m}_{k-1}$, and the Jacobian $\mathbf{H}_\mathbf{x}$ of $\mathbf{h}(\mathbf{x})$ is evaluated at $\mathbf{x} = \mathbf{m}_k^-$.
```

Derivation[7] We compute the first order Taylor approximation of $\mathbf{f}$ around $\mathbf{x} = \mathbf{m}_{k-1}$:

$$\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\mathbf{m}_{k-1}) + A_k(\mathbf{x}_{k-1} - \mathbf{m}_{k-1}) \tag{37}$$

And the first order Taylor approximation of $\mathbf{h}$ around $\mathbf{x} = \mathbf{m}_k^-$:

$$\mathbf{h}(\mathbf{x}_k) \approx \mathbf{h}(\mathbf{m}_k^-) + C_k(\mathbf{x}_k - \mathbf{m}_k^-) \tag{38}$$

where

$$\begin{aligned} A_k &= \nabla_\mathbf{x}\mathbf{f}\,|_{\mathbf{m}_{k-1}}, \\ C_k &= \nabla_\mathbf{x}\mathbf{h}\,|_{\mathbf{m}_k^-} \end{aligned} \tag{39}$$

By substituting (37) and (38) into (31) and (32) respectively, we obtain a linear Gaussian state space model:

$$\begin{aligned} \mathbf{x}_k &= a_k + A_k\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= c_k + C_k\mathbf{x}_k + \mathbf{r}_k \end{aligned} \tag{40}$$

where

$$\begin{aligned} a_k &= \mathbf{f}(\mathbf{m}_{k-1}) - A_k\mathbf{m}_{k-1}, \\ c_k &= \mathbf{h}(\mathbf{m}_k^-) - C_k\mathbf{m}_k^- \end{aligned} \tag{41}$$

We can now run the Kalman filter on this linearised system (40); doing so obtains the prediction and update equations as shown in (35) and (36).

Algorithm 2 First order EKF filter for systems with non-additive noise[1]

```
1: Prediction:
$$\mathbf{m}_k^- = \mathbf{f}(\mathbf{m}_{k-1}, \mathbf{0}),$$
$$\mathbf{P}_k^- = \mathbf{F_x}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{F_x}^\top(\mathbf{m}_{k-1}) + \mathbf{F_q}(\mathbf{m}_{k-1})\mathbf{Q}_{k-1}\mathbf{F_q}^\top(\mathbf{m}_{k-1}) \quad (42)$$
2: Update:
$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-, \mathbf{0}),$$
$$\mathbf{S}_k = \mathbf{H_x}(\mathbf{m}_k^-)\mathbf{P}_k^-\mathbf{H_x}^\top(\mathbf{m}_k^-) + \mathbf{H_r}(\mathbf{m}_k^-)\mathbf{R}_k\mathbf{H_r}^\top(\mathbf{m}_k^-),$$
$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H_x}^\top(\mathbf{m}_k^-)\mathbf{S}_k^{-1},$$
$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k,$$
$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top \quad (43)$$
   where $\mathbf{F_x}(\mathbf{m})$ is the Jacobian of $\mathbf{f}$ with respect to state, $\mathbf{F_q}(\mathbf{m})$ is the Jacobian of $\mathbf{f}$ with respect to noise, $\mathbf{H_x}(\mathbf{m})$ is the Jacobian of $\mathbf{h}$ with respect to state, and $\mathbf{H_q}(\mathbf{m})$ is the Jacobian of $\mathbf{h}$ with respect to noise.
```

Derivation[6] We follow a similar derivation to the derivation of **Algorithm 1**. Compute the first order Taylor approximation of $\mathbf{f}$ around $(\mathbf{m}_{k-1}, \mathbf{0})$:

$$\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{q}_{k-1}) \approx \mathbf{f}(\mathbf{m}_{k-1}, \mathbf{0}) + A_k(\mathbf{x}_{k-1} - \mathbf{m}_{k-1}) + B_k\mathbf{q}_{k-1} \tag{44}$$

And the first order Taylor approximation of $\mathbf{h}$ around $(\mathbf{m}_k^-, \mathbf{0})$:

$$\mathbf{h}(\mathbf{x}_k, \mathbf{r}_k) \approx \mathbf{h}(\mathbf{m}_k^-, \mathbf{0}) + C_k(\mathbf{x_k} - \mathbf{m_k^-}) + D_k\mathbf{r}_k \tag{45}$$

where

$$\begin{aligned} A_k &= \nabla_{\mathbf{x}}\mathbf{f}\ |_{(\mathbf{m}_{k-1}, \mathbf{0})}, \\ B_k &= \nabla_{\mathbf{q}}\mathbf{f}\ |_{(\mathbf{m}_{k-1}, \mathbf{0})}, \\ C_k &= \nabla_{\mathbf{x}}\mathbf{h}\ |_{(\mathbf{m}_k^-, \mathbf{0})}, \\ D_k &= \nabla_{\mathbf{r}}\mathbf{h}\ |_{(\mathbf{m}_k^-, \mathbf{0})} \end{aligned} \tag{46}$$

We substitute (44) and (45) into (33) and (34) to obtain the linear Gaussian model:

$$\begin{aligned} \mathbf{x}_k &= a_k + A_k\mathbf{x}_{k-1} + B_k\mathbf{q}_{k-1}, \\ \mathbf{y}_k &= c_k + C_k\mathbf{x}_k + D_k\mathbf{r}_k \end{aligned} \tag{47}$$

where

$$\begin{aligned} a_k &= \mathbf{f}(\mathbf{m}_{k-1}, \mathbf{0}) - A_k \mathbf{m}_{k-1}, \\ c_k &= \mathbf{h}(\mathbf{m}_k^-, \mathbf{0}) - C_k \mathbf{m}_k^- \end{aligned} \tag{48}$$

Once again, we obtain the prediction and update equations shown in (42) and (43) by applying the Kalman filter on this linearised system (47).

Table 1: Advantages vs. Disadvantages of EKF

Advantages	**Disadvantages**
Applicable to a wider range of systems than Kalman Filter	Requires noise to be Gaussian
Preserves the simplicity of Kalman Filters	Requires dynamic and measurement models to be differentiable
Computationally efficient[1]	Linearisation based on only a local region around mean

4.2 Summary

So far we have applied first order Taylor series approximations to produce EKF algorithms for non-linear systems. Such an approach, using second order Taylor series approximations may also be applied to produce second order EKF algorithms for systems with both additive and non-additive noise. But what if we come across systems where approximating the filtering distributions as Gaussian is not suitable? In the next section, we discuss another filtering method which can provide more suitable approximations.

5 Particle Filter

5.1 Background

Particle Filters are Monte Carlo algorithms which are used to solve filtering problems, applicable from signal processing to statistical inference. Again, we aim to find the posterior distributions of states in a Markov process using just noisy observations for input. The term ”particle filtering” stems originally from the link to fluid mechanics and particle methods used in that field. Whilst in a linear system, the Kalman filter is optimal, when there is non-Gaussian noise, the Particle Filter often performs better[8]. It is also more appropriate if the filtering distribution is multimodal or if some state components are discrete[9].

5.2 Monte Carlo and Importance Sampling

Monte Carlo refers to methods where closed form computation of certain quantities is now replaced by taking samples from the distribution and estimating the quantities with sampling averages. Considering a distribution on X with density π with respect to dx, we can express $\pi(x)$ as $\frac{\gamma(x)}{Z}$ where Z is a normalising constant, so that γ is a non normalised density[6]. We aim to work out integrals like $\pi(\phi) = E_{\pi}[\Phi(x)] = \int x\phi(x)\pi(dx)$ and in most cases using Monte Carlo methods we need to be able to compute γ point by point.
In the case where we are able to obtain N i.i.d. samples $X_i \sim \pi$, it is possible to make the following perfect Monte Carlo approximation to $\pi(\phi)$:

$$\widehat{\pi}(\phi) = \int_X \phi(X)\widehat{\pi}(dx) = \frac{1}{N}\sum_{i=1}^{N}\phi\left(X^i\right) \tag{49}$$

We can interpret $\widehat{\pi}(dx)$ as $\frac{1}{N}\sum_{i=1}^{N}\delta_{X^i}(dx)$ where delta is the Dirac measure i.e. $\int_{x'}(A) = 1x' \in A$[6].
The estimator $\widehat{\pi}(\phi)$ is consistent, using the Strong Law of Large Numbers, it is unbiased and has a variance that decreases linearly with N. These results are shown below:

$$E^N\left[\sum_{i=1}^{N}\phi\left(X^i\right)\right] = \sum_{i=1}^{N}E_{\pi}\left[\phi\left(X^i\right)\right] = NE_{\pi}[\phi(X)) \tag{50}$$

$$Var^N\left[\widehat{\pi}(\phi)\right] = \frac{1}{N}Var_{\pi}\left[\phi\left(X^i\right)\right] = \frac{1}{N}\left(\pi\left(\phi^2\right) - \pi(\phi)^2\right) \tag{51}$$

hence $E^N\left[\sum_{i=1}^{N}\phi\left(X^i\right)\right] = \pi(\phi)$ where $E^N[\cdot]$ is the expectation with respect to the sampling distribution. Similarly, $Var^N[\cdot]$ is the variance with respect to the sampling distribution.

Particle Filtering relies on Sequential Importance Sampling hence we first consider the simpler version: Importance Sampling. Importance Sampling gets around the inability to sample directly for π, using that $\pi(x) = \frac{w(x)q(x)}{\int w(x)q(x)dx}$ where $w(x) = \frac{\gamma(x)}{q(x)}$. q is the importance distribution while w is the function of unnormalised importance weights. We consider the case where Z in unknown, which is the most common case. In this case, if we can obtain N i.i.d. samples $X_i \sim q$, then $\pi(\phi)$ can be approximated as:

$$\widehat{\pi}(\phi) = \sum_{i=1}^{N} W^i \phi\left(X^i\right) \tag{52}$$

where $W^i = \frac{w\left(X^i\right)}{\sum_{j=1}^{n} w\left(X^j\right)}$. This ensures that $\sum_{i=1}^{N} W^i = 1$ and this is referred to as self-normalising Importance Sampling. Then as with perfect Monte Carlo except with weighting this time:

$$\widehat{\pi}(dx) = \sum_{i=1}^{N} W^i \delta_{X^i}(dx) \tag{53}$$

We can also estimate the unknown Z as:

$$Z = \int \frac{\gamma(x)}{q(x)} \widehat{q}(dx) = \frac{1}{N} \sum_{i=1}^{N} W\left(X^i\right) \widehat{q}(dx) = \frac{1}{N} \sum_{i=1}^{N} \delta_{X^i}(dx) \tag{54}$$

$\widehat{Z}$ is constructed using perfect Monte Carlo, therefore it is unbiased, while the estimate for $\pi(\phi)$ is not necessarily unbiased, however it is still asymptotically unbiased and Central Limit Theorem holds. Again, the rate of decrease of the variance is $1/N$:

$$Var^N\left(\widehat{Z}\right) = \frac{Z^2}{N}\left(\int \frac{\pi^2(x)}{q(x)} dx - 1\right) \tag{55}$$

The ideal value for q in order to to minimise the variance of the estimator to $\pi(\phi)$ is:

$$q(x) = \frac{\left|\phi(x)\right| \pi(x)}{\int_X |\phi(x)| \pi(x) dx} \tag{56}$$

However it is not very useful and difficult to work out in practise. Another useful way to choose q is to try to minimise the variance of $\widehat{Z}$ which is equivalent to minimising the variance of the importance weights. Both result in $q = \pi$.

The Effective Sample Size $ESS = \frac{N}{1 + Var_q\left(\frac{w(X)}{Z}\right)}$ is a measure of the efficiency of the model with the given q which corresponds to how many samples using perfect Monte Carlo would result in the same variance as obtained[6].

5.3 Sequential Importance Sampling

It is the sequential aspect of this method that allows for both discrete and continuous components. If the target density is of the form $\pi(x_{0:T}) = \frac{\gamma(x_{0:T})}{Z}$, i.e. a sequence of distributions, we can implement Importance Sampling recursively.
Hence, for the sequence $\{\pi_n(x_{0:T})\}_{n\leq T}$ with $\pi = \pi_T$, we assume that there exists r such that:

$$\gamma_n(x_{0:n}) = \gamma_{n-1}(x_{0:n-1})\, r_n(x_n|x_{0:n-1}) \tag{57}$$

We define each next proposal as:

$$q_n(x_{0:n}) = q_{n-1}(x_{0:n-1})\, q_n(x_n|x_{0:n-1}) \tag{58}$$

and also recursively define the weights using:

$$w_n(x_{0:n}) = w_{n-1}(x_{0:n-1})\frac{r_n(x_n|x_{0:n-1})}{q_n(x_n|x_{0:n-1})} \tag{59}$$

We estimate π_n and Z_n as:

$$\widehat{\pi}_n(dx_{0:n}) = \sum_{i=1}^{N} W_n^i \delta_{X_{0:n}^i}(dx_{0:n}) \tag{60}$$

$$\widehat{Z}_n = \frac{1}{N}\sum_{i=1}^{N} w_n\left(X_{0:n}^i\right) \tag{61}$$

Alternatively, we can define the estimator using:

$$\widehat{Z}_n = \prod_{k=0}^{N}\frac{\widehat{Z}_k}{Z_{k-1}} \tag{62}$$

$$\frac{\widehat{Z}_n}{Z_{n-1}} = \sum_{i=1}^{N} W_{n-1}^i \frac{r_k\left(X_n^i|X_0^i : n-1\right)}{q_k\left(X_n^i|X_0^i : n-1\right)} \tag{63}$$

We show below the algorithm for SIS.

5.4 Sequential Importance Resampling

Sequential Importance Resampling is the next idea used in the Particle Filtering method after SIS. The need to resample arises from the weight degeneracy that we get for large n, whereby low weights remain low and the mass concentrates on a few particles. Almost all weights are either zero or near zero. In short, resampling removes low weights and selects particles with high weights,duplicating these, hence redistributing the weight budget among these[10].
The algorithm for SIR when used in particle filtering is as shown. There are several possible resampling methods and how appropriate these are depends on the application. While in theory the distribution remains intact, resampling introduces variance

Algorithm 3 Sequential Importance Sampling as used in filtering

```
When $n = 0$:
sample from $X_0^i \sim q_\theta(x_0|y_0)$
calculate the weights $w_0\left(X_0^i\right)$ and make $W_0^i$ proportional to these, such that $\sum_{i=1}^N W_n^i = 1$
When $n >= 1$:
sample from $X_n^i \sim q_\theta\left(x_n|y_n, X_{n-1}^i\right)$ and let $X_{0:n}^i = \left(X_{0:n-1}^i, X_n^i\right)$
calculate $w_n\left(X_{n-1:n}^i\right)$ and make $W_n^i$ proportional to $W_{n-1}^i w_n\left(X_{n-1:n}^i\right)$ such that $\sum_{i=1}^N W_n^i = 1$
```

into the system and choosing the best method involves minimising this variance. Stratified resampling is the optimal method in terms of variance. One method is to resample only when it is necessary, for example every k time steps. Adaptive resampling resamples at time steps only when the effective number of particles is much lower than the total number of particles.

$$n_{eff} \approx \frac{1}{\sum_{i=1}^N (w_k^i)^2} \tag{64}$$

Resampling is only done when the effective number of particles $n_{eff} < N/10$, if N is the total number of particles. Another such method is multinomial resampling, which uses the multinomial distribution which generalises the binomial distribution. For example, it models the probability of each number of counts for every side of a k-sided die when rolled N times.

Algorithm 4 Sequential Importance Resampling as used in filtering

```
When $n = 0$:
sample from $X_0^i \sim q_\theta(x_0|y_0)$
calculate the weights $w_0\left(X_0^i\right)$ and make $W_0^i$ proportional to these, such that $\sum_{i=1}^N W_n^i = 1$
resample $\{W_0^i, X_0^i\}$ and get $N$ particles $\left\{\frac{1}{N}, \overline{X_0^i}\right\}$ weighted equally
When $n >= 1$:
sample from $X_n^i \sim q_\theta\left(x_n|y_n, \overline{X_{n-1}^i}\right)$ and let $X_{0:n}^i = \left(\overline{X_{0:n-1}^i}, X_n^i\right)$
calculate $w_n\left(X_{n-1:n}^i\right)$ and make $W_n^i$ proportional to $w_n\left(X_{n-1:n}^i\right)$ such that $\sum_{i=1}^N W_n^i = 1$
resample $\{W_n^i, X_{0:n}^i\}$ and get $N$ particles $\left\{\frac{1}{N}, \overline{X_{0:n}^i}\right\}$ weighted equally
```

5.5 Bootstrap filter

The Bootstrap filter is a variant of the SIR model where the dynamic model is taken to be the importance distribution[11]. In this case the implementation is easy due to

Algorithm 5 Multinomial Resampling

```
draw the sample $(o_n(1),\dots,o_n(N)) \sim Multinomial\left(N; W^1_{n,\dots,} W^N_n\right)$
copying particles:
for i=1:N do
    if $o_n(i) > 0$ then
        for $j = 1 : o_n(i)$ do
            $X^k_{0:n} = X^i_{0:n}$;
            $k \leftarrow k + 1$
```

the assumption however the importance distribution is so inefficient it may requires a number of Monte Carlo samples to get accurate results by estimating. The resampling is usually done at each time step. As before we sample new points from the dynamic model, compute the weights, normalise them so they sum to infinity and finally resample. Repeat for each timestep.

6 Application of Filters

State space models are widely used in many applications. Estimating the unknown state based on the available measurements is an important and a well studied subject in the literature. Depending on the nature of the problem, these models could involve simple linear equations or complex nonlinearities. Filtering is a general statistical approach for estimating the evolving unknown states from the noisy measurements.
All simulations and results of Kalman Filters and particle filters in the following parts are published in the GitHub repository: https://github.com/Yqcca/Filters.

6.1 Linear Systems

6.1.1 Problem Formulation

Assume we have a linear Gaussian state space model:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}_{k-1}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{r}_k \end{aligned} \tag{65}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state, $\mathbf{A}_{k-1}$ is the transition matrix of the dynamic model, $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ is the process noise, $\mathbf{y}_k \in \mathbb{R}^m$ is the measurement, $\mathbf{H}_k$ is the measurement model matrix, and $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0}, \mathbf{R}_k)$ is the measurement noise, given a prior distribution $\mathbf{x}_0 \sim \mathrm{N}(\mathbf{m}_0, \mathbf{P}_0)$.

Car Tracking Model
Consider a target tracking problem, which refers to determining the position or velocity of a remote target, given noisy measurements of the targets position.

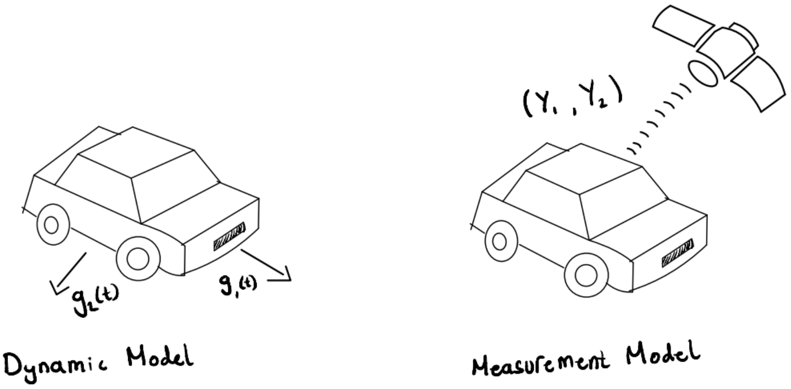


Figure 5: Illustration of the car's dynamic and measurement models. The forces $g_1(t)$ and $g_2(t)$ are white noise processes and (y_1, y_2) are noisy measurements of the car's position.

Specifically, we can build a state space model of a car based on Newton's law $g(t) = ma(t)$, where $a(t)$ is the acceleration, m is the mass of the car, and $g(t)$ is a vector of forces acting on the car. Also we model the acceleration of the car, $a(t) = g(t)/m$, as a two-dimensional white random process:

$$\frac{d^2x_1}{dt^2} = w_1(t),$$
$$\frac{d^2x_2}{dt^2} = w_2(t) \tag{66}$$

In this case, $x_1(t)$ and $x_2(t)$ determines to the position of the car and $x_3(t) = \frac{dx_1}{dt}$, $x_4(t) = \frac{dx_2}{dt}$ determines the velocity of the car. We can now write this model as a first order system of differential equations:

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \tag{67}$$

In reality, we can only measure the position of the car in discrete time, so we can instead consider the discretized version of the dynamic model, with a sampling period of Δt:

$$\begin{pmatrix} x_{1,k} \\ x_{2,k} \\ x_{3,k} \\ x_{4,k} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta\text{t} & 0 \\ 0 & 1 & 0 & \Delta\text{t} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_{1,k-1} \\ x_{2,k-1} \\ x_{3,k-1} \\ x_{4,k-1} \end{pmatrix} + \mathbf{q}_{k-1} \tag{68}$$

where $\mathbf{q}_{k-1}$ is a discrete-time Gaussian noise process[1]. Hence our dynamical model is of the form

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{q}_{k-1}$$

Our measurements of the cars position (x_1, x_2), under this model, are also subject to Gaussian measurement noise ($r_{1,k}$, $r_{2,k}$).

$$y_{1,k} = x_{1,k} + r_{1,k} \tag{69}$$
$$y_{2,k} = x_{2,k} + r_{2,k} \tag{70}$$

Hence our measurement model is:

$$\mathbf{y}_k = \mathbf{H}\mathbf{x}_k + \mathbf{r}_k, \quad \mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{71}$$

Our noise terms are $\mathbf{q}_{k-1} \sim \text{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{r}_k \sim \text{N}(\mathbf{0}, \mathbf{R})$ where

$$\mathbf{Q} = \begin{pmatrix} \frac{q_1^c\Delta t^3}{3} & 0 & \frac{q_1^c\Delta t^2}{2} & 0 \\ 0 & \frac{q_2^c\Delta t^3}{3} & 0 & \frac{q_2^c\Delta t^2}{2} \\ \frac{q_1^c\Delta t^2}{2} & 0 & q_1^c\Delta t & 0 \\ 0 & \frac{q_2^c\Delta t^2}{2} & 0 & q_2^c\Delta t \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{pmatrix}$$

where σ_1^2 and σ_2^2 are measurement noises in each coordinate, and q_1^c and q_2^c are the spectral densities of the process noise in each direction[1].
Our model satisfies the linear Gaussian state space model **(21)** and thus, we can apply the appropriate filters derived beforehand, to estimate the states of the system.

6.1.2 Algorithm Implementation

Before we can apply the relevant linear filters to the system, we first need to generate data for the underlying states of the system $\mathbf{x}_{1:T}$ and the measurement data $\mathbf{y}_{1:T}$. We simulate the motion of the car, based on equation **(59)** and the noisy measurement of the car's position using **(62)**. We do this for $N = 100$ time steps, with $\Delta t = 0.1$ and set $q_1^c = q_2^c = 1$ and $\sigma_1^2 = \sigma_2^2 = 1/2$.

Kalman Filter
To apply the Kalman filter, we use **Proposition 3.1** recursively, with the matrices and vectors defined in the problem formulation of the Car Tracking Model in the previous section. Our initial mean $\mathbf{m}_0 = (0, 0, 0, 0)^T$ and initial covariance matrix $\mathbf{P}_0 = \mathbf{Q}$. We use the algorithm recursively applying **predict** and **update** steps in **algorithm (number)** $N = 100$ times to get our filtered states using Kalman Filter.

Particle Filter
To apply the particle filters to linear Gaussian problem, we firstly need to determine two **hyper-parameters** of the particle filter:

1) Number of particles N

2) Proposal distribution

1) In the following part, we will run a simulation and visualize how different number of particles of bootstrap particle filters influence the estimation in the car tracking model under controlled proposal distribution. Figure 6 shows the true states, measurement and estimated trajectories $E(X_k|Y_{0:k})$.

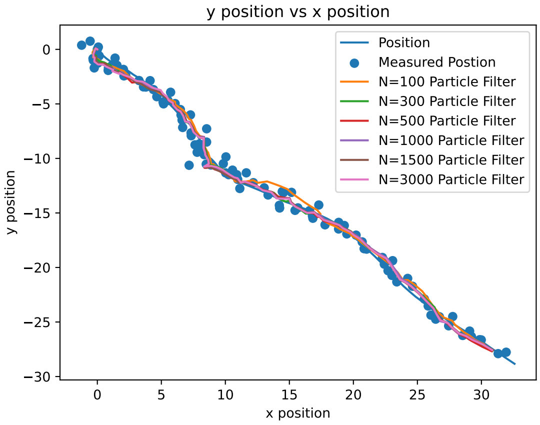
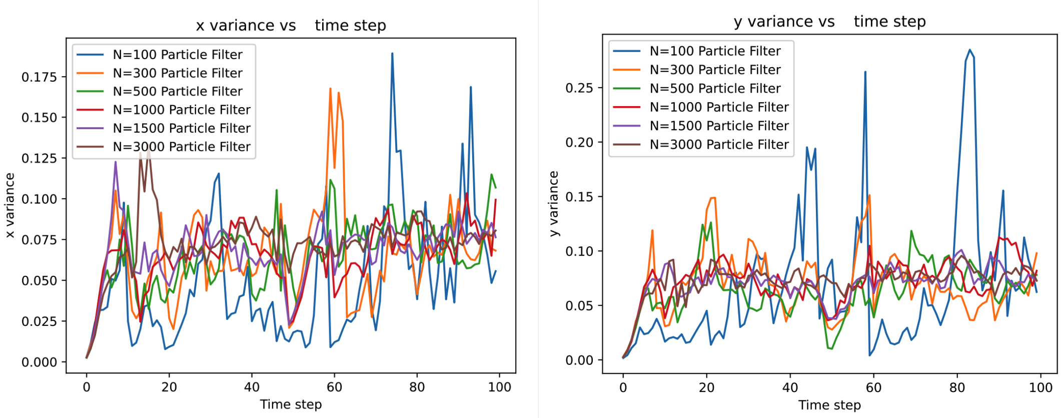


Figure 6: Car tracking using bootstrap filters with different number of particles

We separate the trajectories above into x and y directions. Fig.7(a) is the estimated x-position $E(x_k|Y_{0:k})$, and Fig.7(b) is the estimated y-position $E(y_k|Y_{0:k})$.



(a) x-variance vs time step

(b) y-variance vs time step

Figure 7: Variance of estimation of different bootstrap filters at each time step

In the car tracking simulation, the mean square error is calculated by $\sum_{k=1}^{t} \frac{(E(x_k|Y_{0:k})-x_k)+(E(y_k|Y_{0:k})-y_k)}{2*t}$, where $E(x_k|Y_{0:k})$ is the estimated hidden state in x-direction, $E(y_k|Y_{0:k})$ is the estimated hidden state in y-direction, t is the number of time step and x_k, y_k is the true hidden state at time step k. The variance of estimation at time step k is calculated by $\frac{1}{N}\sum_{i=1}^{N} w_k^i(E(X_k|Y_{0:k})-X_k^i)^2)$, where N is the number of particles, X_k^i and w_k^i are the samples and sample weights at time step k.

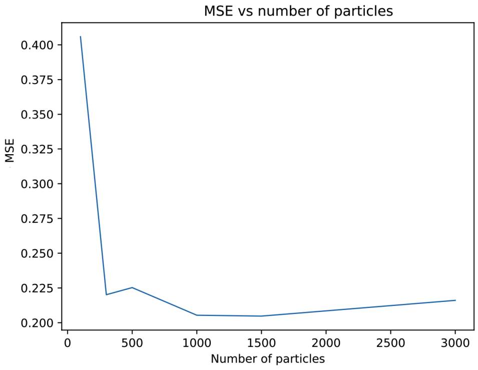


Table 2: MSE vs number of particles

Number of particles	MSE
N = 100	0.405875
N = 300	0.220143
N = 500	0.225257
N = 1000	0.205376
N = 1500	0.204754
N = 3000	0.216068

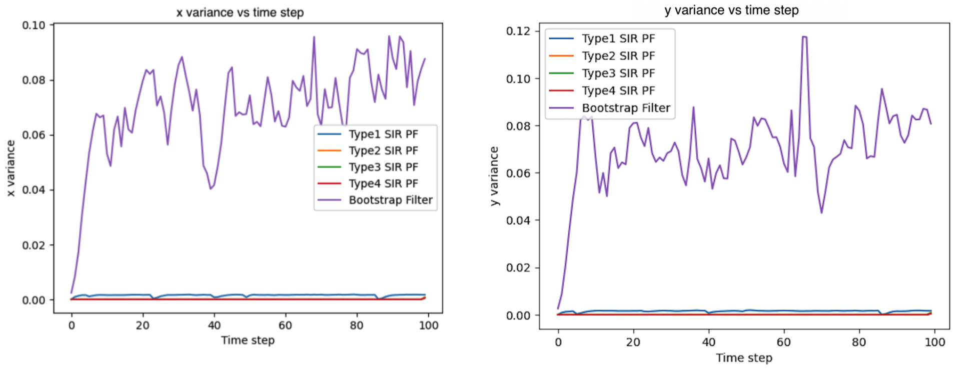
Figure 8: MSE vs Number of particles

Determining the number of particles is the trade-off between **computational cost** and **mean square error** of the resulting estimates. According to the figure and tabular above, the trend indicates that as we increase the number of particles, the former increases, while the latter decreases. In addition, the greater number of particles, the variance of estimations is more stable. We set the number of particles $N = 1000$ in the following simulations to get our filtered state.

2) As in standard importance resampling, the key to designing well performing algorithms is to find a good proposal q. The proposal distribution should be in a functional form that we can easily draw samples from it and that it is possible to evaluate the probability densities of the sample points.

In our implementation of SIR particle filter, we use the **weighted sum** of the sample AX_{k-1} and the nearest measurement Y_k as the mean of proposal distribution. We only consider the nearest measurement Y_k, as opposed to the previous measurement $Y_{0:k}$, since it leads to the optimal proposal in terms of variance. In addition, we use **covariance matrix of dynamic model** as the covariance of proposal distribution. Due to computation cost limitation, we will only consider the proposal distributions with different mean. In addition, one simple and popular option is to use the bootstrap proposal mentioned in Chapter6. This proposal works well when y_k being not very informative (i.e. having a fairly flat likelihood function).

In the following part, we will run a simulation and compare different proposal distribution with different mean and bootstrap distribution under controlling the number of particles $N = 1000$.



(a) x-variance vs time step (b) y-variance vs time step

Figure 9: Variance of estimations of different SIR particle filters

Table 3: MSE vs types of proposals

The mean of proposal distribution	MSE
Type1 : $\mu = 0.5 * AX_{k-1} + 0.5 * Y_k$	0.319604
Type2 : $\mu = 0.4 * AX_{k-1} + 0.6 * Y_k$	0.265479
Type3 : $\mu = 0.3 * AX_{k-1} + 0.7 * Y_k$	0.237944
Type4 : $\mu = 0.2 * AX_{k-1} + 0.8 * Y_k$	0.232609
Type5 : Bootstrap proposal	0.150585

According to the table above, we can derive the optimal proposal distribution among those proposals for car tracking example is the Type5:Bootstrap proposal, corresponding to the least mean square error. Although bootstrap filter has higher variance than that of other particle filters, the accuracy of its estimation is the best among those proposal distribution candidates. Among weighed sum method proposals, Type2 perform the best, since it sets a proper weight between the dynamic state and the observation. If the observation includes little noise, we should weight more on the observation; otherwise, we should weight more on the dynamic state. Since the measurement model are linear Gaussian with remarkable noise, the bootstrap proposal option is a potential optimal proposal distribution for SIR particle filter. We use Type2 proposal distribution for SIR filters and Type1 proposal distribution for SIS filter to conduct following simulations.

6.1.3 Numerical Results

After determining two hyper-parameters for each particle filter: number of particles and proposal distribution, we can apply Kalman Filter and our designed particle filters. In the following part, we will use SIR particle filter, adaptive SIR particle filter, Bootstrap particle filter, SIS particle filter, Kalman Filter to estimate the car trajectory, according to the problem mentioned in the Section 7.1.1.

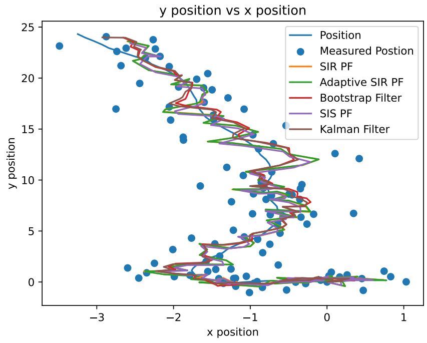
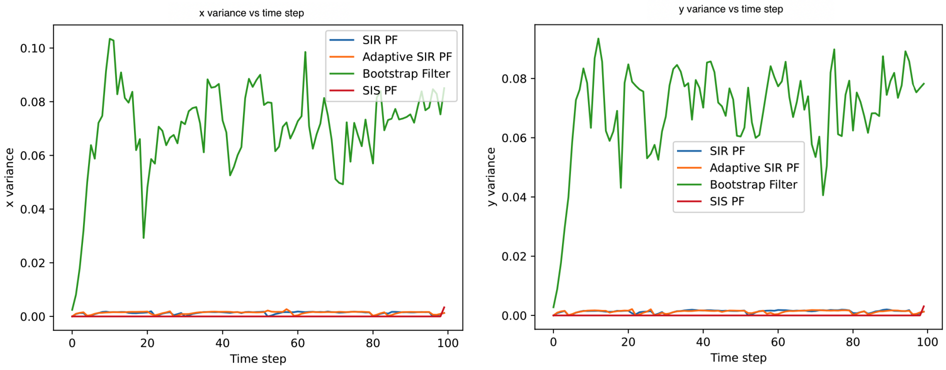


Figure 10: Car tracking by using different particle filters

The Figure 10 exhibits the path of car tracking trajectories under different filters $E(X_k|Y_{0:k})$ and 11 shows the variance of estimations at each time step. We can see the variance of bootstrap filter is higher than that of other particle filter.



(a) x-variance vs time step

(b) y-variance vs time step

Figure 11: Variance of estimation at each time step

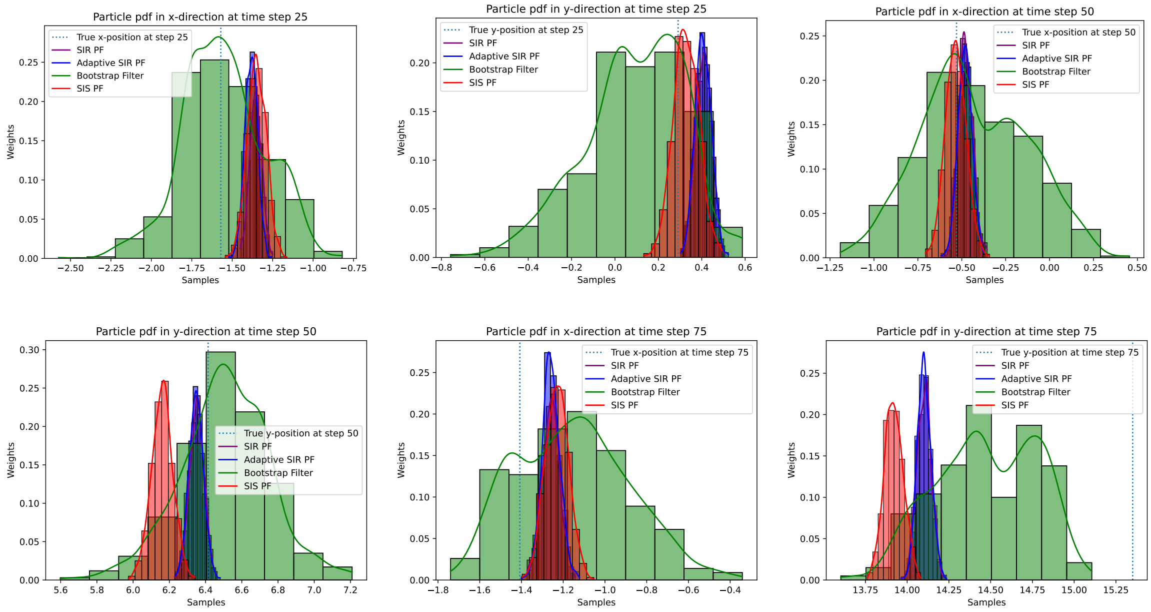


Figure 12: pdf of estimations of different particle filters at time step $t = 25, 50, 75$

Based on weights and samples, we can derive the the filtering distribution for estimation at time step k by $p(x_k|y_{1:k}) \approx \sum_{i=1}^{N} \delta(x_k - x_k^i)$. As shown in Figure.12 and Table.4, we can compare the performance of each filters in this simulation quantitatively based on the probability density functions and variance given by the true motion trajectory and the measurement. Based on Figure.12, we can see the probability density function of estimation at x-position and y-position at time step $25, 50, 75$.

Particle filtering uses a set of particles to represent the posterior distribution. As shown in Figure.12, we can gauge the likely outcome of those particle values based on the filtering distribution. According to the probability density function, we can figure that the true hidden state lie in which percentage confidence interval based on our particle filter estimations. The confidence interval is a range of estimates for an unknown parameter. According to the figures above, almost all true hidden states lie in the range of the bootstrap particle filter filtering distribution. Specifically, at time step 25 and 50, the true hidden states lie in 90% confidence interval of bootstrap filter filtering distribution. It implies the bootstrap filter can estimate the true hidden state well, since it has the optimal proposal in this linear Gaussian noise case mentioned in Section 6.1.2. In addition, although the variance of other particle filters is low, which means high precision of our filtering estimations, it may lead to the situation that the true hidden state not lie in the confidence interval of those filtering distribution. In other words, it is possible to have a nice narrow filtering distribution, but the mean of that distribution is centered at the wrong place because of some other errors, such as the error caused by the proposal distribution.

Table 4: MSE vs types of filters

Types of particle filters	MSE
SIR PF (N=1000)	0.311513
SIR adaptive PF (N=1000)	0.310121
Bootstrap filter (N=1000)	0.549897
SIS PF (N=1000)	0.339509
Kalman filter	0.121171

Firstly, we can note **Kalman Filter** has the lowest mean square error, which means its estimations are the most accurate. In linear Gaussian noise model, it can be interpreted as a benchmark for particle filters.

SIS particle filter is on the top among filters and has the highest mean square error, since it has degeneracy problem mentioned in Chapter6, influencing the accuracy of the estimations. Particle filters with resampling procedure (**SIR particle filter, adaptive SIR particle filter**) have less mean square error than SIS particle filter and their estimations are acceptable compared to Kalman Filter. Although they have almost the same performance in estimations, adaptive SIR particle filter has less computation cost than SIR particle filter due to its adaptive mechanism. Stated in the previous section, because bootstrap proposal is a potential optimal proposal distribution, bootstrap filter performs the best among those particle filters, giving us the most accurate estimations among those particle filters.

To summarize, in linear Gaussian case, Kalman Filter can give us the most accurate estimations and the benchmark for particle filters. Considering particle filter, applying resampling method can solve degeneracy problem of particle filters. In addition, a well designed proposal distribution is crucial. Proper proposal distribution of particle filter can give us a huge improvement in estimation accuracy.

6.2 Non-Linear Systems

6.2.1 Problem Formulation

The general, non-linear state space model, with non-additive noise is of the form:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1},\ \mathbf{q}_{\mathbf{k-1}}), \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k,\ \mathbf{r}_k), \end{aligned} \tag{72}$$

In the following problems, we will consider a more specific non-linear state space model, with additive noise which is of the form:

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{q}_{k-1}, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{r}_k, \end{aligned} \tag{73}$$

where the function $\mathbf{f}$ is the dynamic model function and the function $\mathbf{h}$ is the measurement model function. These functions can be dependent on the step number k. Here, $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0},\ \mathbf{Q}_{k-1})$ is the process noise, and $\mathbf{r}_k \sim \mathrm{N}(\mathbf{0},\ \mathbf{R}_k)$ is the measurement noise.

Noisy Pendulum Model
We will consider a simple pendulum model with unit mass and length that is subject to a random noise process.

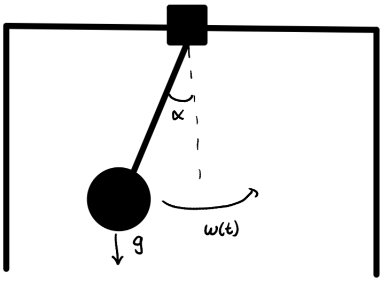


Figure 13: Pendulum illustration. The forces on the pendulum are gravity g and a white noise force w(t)

The governing equation for the dynamics of this model is the following differential equation

$$\frac{d^2\alpha}{dt^2} = -g\sin\alpha + w(t) \tag{74}$$

where α is the angle the pendulum makes with the vertical, g is the gravitational constant and $w(t)$ is a random noise process[1]. This second order differential equation can be written as a system of first order differential equations, and we get the following state space model:

$$\frac{d}{dt}\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}x_2\\-g\sin x_1\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix}w(t), \tag{75}$$

where $x_1 = \alpha$ and $x_2 = \frac{d\alpha}{dt}$. Assume now that we can only measure the horizontal position of the pendulum. This would result in a non-linear measurement model

$$y_k = \sin\alpha(t_k) + noise \tag{76}$$

Now if we discretize this pendulum model, we get the following:

$$\begin{pmatrix}x_{1,k}\\x_{2,k}\end{pmatrix} = \begin{pmatrix}x_{1,k-1}+x_{2,k-1}\Delta t\\x_{2,k-1}-g\sin(x_{1,k-1})\Delta t\end{pmatrix} + \mathbf{q}_{k-1}, \tag{77}$$

$$y_k = \sin(x_{1,k}) + r_k, \tag{78}$$

where $\mathbf{q}_{k-1} \sim \mathrm{N}(\mathbf{0},\ \mathbf{Q})$ and $r_k \sim \mathrm{N}(0,\ R)$ and

$$\mathbf{Q} = \begin{pmatrix}\frac{q^c\Delta t^3}{3} & \frac{q^c\Delta t^2}{2}\\ \frac{q^c\Delta t^2}{2} & q^c\Delta t\end{pmatrix}, \quad R = \begin{pmatrix}\sigma^2\end{pmatrix} \tag{79}$$

where q^c is the spectral density of the continuous-time process noise[1]. The model now follows the non-linear state space model with additive noise, with

$$\mathbf{f}(\mathbf{x_{k-1}}) = \begin{pmatrix}x_{1,k-1}+x_{2,k-1}\Delta t\\x_{2,k-1}-g\sin(x_{1,k-1})\Delta t\end{pmatrix}, \quad \mathbf{h}(\mathbf{x_{k-1}}) = \sin(\mathbf{x_{1,k}}) \tag{80}$$

6.2.2 Algorithm Implementation Details

As with the linear car tracking problem, to obtain the data of the underlying states of the system $\mathbf{x}_{1:T}$ and the measurement data $\mathbf{y}_{1:T}$, we simulate the motion of the pendulum using **(71)** and simulate the measurements of the horizontal angular position of the pendulum using **(72)**. We simulate the for $N = 100$ time steps with $\Delta t = 0.1$, $\sigma = 1/4$ and $q^c = 1/2$.

Extended Kalman Filter
To estimate the s tate variables, we can use a first order EKF filter for systems with additive noise. The Jacobian of $\mathbf{F}_x$ of **f(x)** and the Jacobian $\mathbf{H}_x$ of **h(x)** for the noisy pendulum system are therefore as follows

$$\mathbf{F}_x(\mathbf{x}) = \begin{pmatrix}1 & \Delta t\\ -g\cos(x_1)\Delta t & 1\end{pmatrix}, \quad \mathbf{H}_x(\mathbf{x}) = \begin{pmatrix}\cos(x_1) & 0\end{pmatrix} \tag{81}$$

Using also the matrices and vectors defined in the problem formulation of the Noisy Pendulum Model, we can apply **Algorithm 1** recursively, applying the **predict** and **update** steps at each time step k on the measured data $\mathbf{y}_k$ to estimate the state of the system at that time step $\mathbf{x}_k$.

Particle Filter
To apply the particle filters to nonlinear Gaussian problem, we firstly need to determine two **hyper-parameters** of the particle filter:

1) Number of particles choice N

2) Proposal distribution

The hyper-parameter determining process is the same as that mentioned in Section 7.1.2. We need to run several simulations and then determine the proper number of particles of particle filter and the optimal proposal distribution. After those simulations, we set the number of particles $N = 1500$ and use proposal distribution $N(0.3 * f(X_{k-1}) + 0.7 * Y_k, Q)$ for SIR filters and $N(0.4 * f(X_{k-1}) + 0.6 * Y_k, Q)$ for SIS particle filter.

6.2.3 Numerical Results

After determining two hyper-parameters for each particle filter: number of particles and proposal distribution, we can apply Extended Kalman Filter and our designed particle filters to the pendulum model and compare their estimations. In the following part, we will use SIR particle filter, adaptive SIR particle filter, Bootstrap particle filter, SIS particle filter, Extended Kalman Filter to estimate the pendulum angle position, according to the pendulum problem mentioned in the Section 7.2.1.

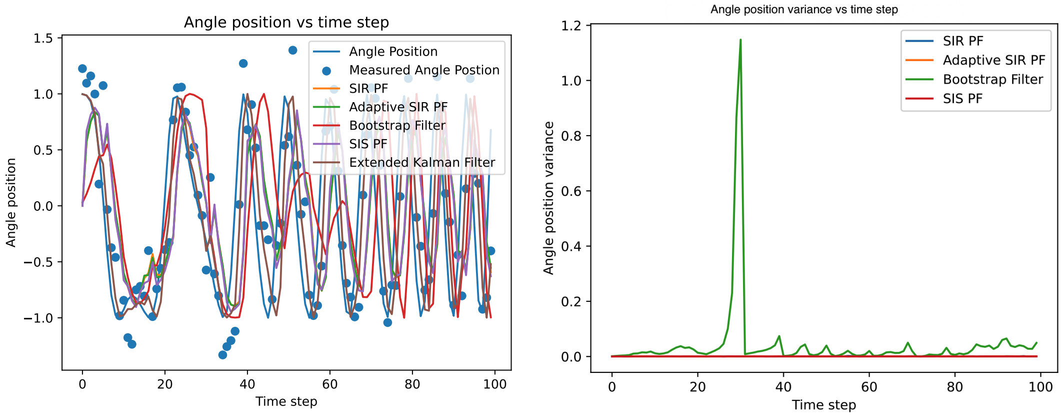


Figure 14: Variance of estimation at each time step

The Figure 14 exhibits the true, measured angle and estimated pendulum angle position $E(X_k|Y_{0:k})$ under different filters.

In the pendulum simulation, the MSE is calculated by $\sum_{k=1}^{t} \frac{E(X_k|Y_{0:k})-X_k}{2*t}$, where $E(X_k|Y_{0:k})$ is the estimated hidden state of angle position, t is the number of time step and X_k is the true angle position at time step k. The variance of estimation at time step k is calculated by $\frac{1}{N}\sum_{i=1}^{N} w_k^i(E(X_k|Y_{0:k}) - X_k^i)^2)$, where N is the number of particles, X_k^i and w_k^i are the samples and sample weights at time step k. In the pendulum example, the variance of all particle filters are almost close to zero.

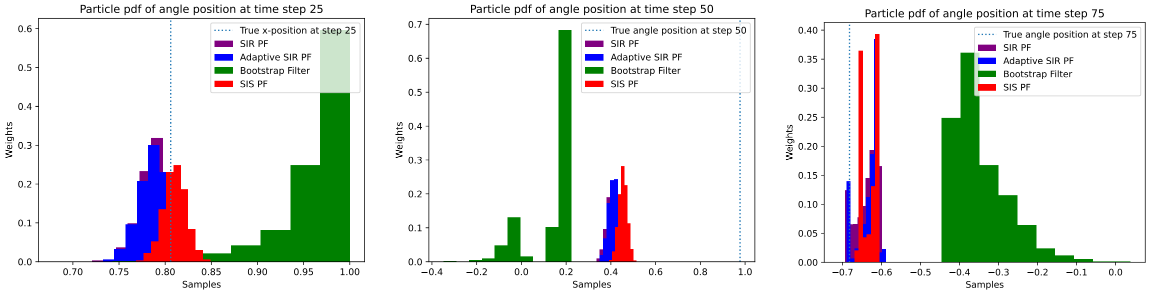


Figure 15: pdf of estimations of different particle filters at time step $t = 25, 50, 75$

Based on weights and samples, we can derive the the filtering distribution for estimation at time step k by $p(x_k|y_{1:k}) \approx \sum_{i=1}^{N} \delta(x_k - x_k^i)$. As shown in Figure.15 and Table.4, we can compare the performance of each filters in this simulation quantitatively based on the probability density functions and variance given by the true angle position and the measurement. Based on Figure.15, we can see the probability density function of estimation at angle position at time step $25, 50, 75$.

By using the same method mentioned in Section 6.1.3, we can derive the filtering distributions for each particle filter. According to the probability density functions, we can figure out the true hidden state lie in which percentage confidence interval based on our particle filter estimations. At time step 25, the true hidden state lies in 95% confidence interval of SIS particle filter filtering distribution and lies in 80% confidence interval of SIR particle filter filtering distribution and SIR adaptive particle filter filtering distribution. At time step 75, the true hidden state is near the one of peaks of particle filters, excluding bootstrap particle filter, probability density distribution. This phenomenon implies those particel filters perform well under the nonlinear Gaussian noise model. In addition, as the true hidden state doesn't lie in the bootstrap filter filtering distribution, the bootstrap filter perform not well in this nonlinear Gaussian noise model, due to the low noise in the measurement model.

In addition, because low noise in this dynamic model causes sample impoverishment, it also implies that pure recursive estimation with particle filters is challenging. This is because in pure recursive estimation the process noise is formally zero and thus a basic SIR based particle filter is likely to perform very badly. The problem can be solved by Rao–Blackwellization, but this method will not be mentioned here due to computation limitation.

Table 5: MSE vs types of filters

Types of particle filters	MSE
SIR PF (N=1500)	0.331764
SIR adaptive PF (N=1500)	0.333994
Bootstrap filter (N=1500)	0.528719
SIS PF (N=1500)	0.352238
Extended Kalman filter	0.114149

Firstly, we can note **Extended Kalman Filter** has the lowest mean square error, which means its estimations perform the best. Since pendulum angle position estimation is a nonlinear Gaussian problem, involving sinusoidal waves in both dynamic and measurement model with a little noise ($\sigma = 0.25$), Extended Kalman Filter approximations (formed by utilizing Taylor series approximations to the non-linearity) can estimate the true angle positions well.

The **bootstrap Filter** has the highest mean square error, since when the observations are quite accurate, the bootstrap approach is not effective. Specifically, mentioned in the previous part, bootstrap filter uses $f_\theta(x_k|x_{k-1})$ as the proposal distribution q_θ. In this nonlinear case, the measurement model only has a little noise, but bootstrap filter cannot guide the particles to areas of higher $p_\theta(x_{0:k}|y_{0:k})$ by using information from y_k, leading to the high variance and bias.

In addition, due to their proper proposal distribution, **SIS particle filter** and particle filters with resampling procedure (**SIR particle filter, adaptive SIR particle filter**) have less mean square error than bootstrap filter and their estimations are acceptable compared to true angle position.

In conclusion, Extended Kalman Filter gives us the most accurate estimations for this nonlinear problem, because Taylor series estimation can well approximate the sinusoidal wave in models. Under the condition that measurement with little noise, bootstrap filter cannot be applied effectively. As mentioned in Section 7.1.3, a well designed proposal distribution is very important for particle filters. In general, the necessity of a good importance proposal holds for IS based methods.

7 Discussion

Bayesian filters have many applications in the empirical world such as in finance, econometric, navigation and much more. In our linear and nonlinear problems (Section 6.1.1, 6.2.1) we explored some real world problems, however due to time and computation cost limitations, we only consider Gaussian noise in both dynamic and measurement models. There can be other non-Gaussian noise in the systems, such as uniform distribution noise.

Gaussian noise is considered in both Kalman and EKF. We have looked at applying EKF algorithms to non-linear systems of the form (31), (32) and (33), (34). Models of this form may also be approximated as Gaussian distributions using an Unscented Kalman Filter. In EKF, we approximated around one mean point, using a Taylor series approximation, whereas in UKF, we approximate around multiple sigma points, using an Unscented Transform.

In the prediction step of UKF, we are given equations to compute the sigma points and their weights, then we transform the sigma points through the dynamic model, and hence calculate predicted mean and covariances based on the weights and transforms. In the update step, we form new sigma points based on the predicted mean and covariance, transform the sigma points, calculate predicted mean, covariance, and cross-covariance, and hence compute the filtered state mean and covariance using filtering equations.[1]

As we have used multiple points, this process requires more computation than EKF, however, it provides a better approximation than using a single point, hence can be more suitable for systems with more extreme non-linearities.

Through our numerical analysis we discovered strengths and weaknesses to each filter. Based on the mean and standard deviation of noise, the Kalman filter can give the optimal linearised estimations. Moreover, it often has a quicker convergence time to the valid state, but as a result of this we experienced larger computational costs. For the particle filter, the key for estimating the hidden states is to choose the optimal proposal distribution.

In addition, our simulations were conducted under several theoretical assumptions. For example, the mass of the car, the string length and the mass of pendulum were all set to be 1. In reality, the estimation problem can become more complex, so if we were to repeat our experiments the algorithms of filters would have to be designed based on those situations in order to guarantee the accuracy of estimations.

8 References

[1] Särkkä, Simo. Bayesian filtering and smoothing. No. 3. Cambridge university press, 2013.

[2] Boyd, Stephen, and Lieven Vandenberghe. Introduction to applied linear algebra: vectors, matrices, and least squares. Cambridge university press, 2018.

[3] Lambert, John. ”What Is State Estimation? And The Bayes Filter”. Johnwlambert.Github.Io, 2018, *https://johnwlambert.github.io/bayes-filter/#state-estimation*.

[4] Chen, Zhe. ”Bayesian filtering: From Kalman filters to particle filters, and beyond.” Statistics 182.1 (2003): 1-69.

[5] Faragher, R., 2012. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. IEEE Signal processing magazine, 29(5), pp.128-132.

[6] Kantas, Nikolas, 2022. MATH70013 - Advanced Simulation Methods. p1-69.

[7] Yu, B., Shenoy, K., Sahani, M., 2004. ”Derivation of Extended Kalman Filtering and Smoothing Equations.” *https://users.ece.cmu.edu/*$\sim byronyu/papers/derive\_eks.pdf$

[8] Djuric, Petar M., et al. ”Particle filtering.” IEEE signal processing magazine 20.5 (2003): 19-38.

[9] Johannes, Michael, and Nicholas Polson. ”Particle filtering.” Handbook of Financial Time Series. Springer, Berlin, Heidelberg, 2009. 1015-1029.

[10] Douc, Randal, and Olivier Cappé. ”Comparison of resampling schemes for particle filtering.” Ispa 2005. proceedings of the 4th international symposium on image and signal processing and analysis, 2005.. IEEE, 2005.

[11] Welch, G.F., 2020. Kalman filter. Computer Vision: A Reference Guide, pp.1-3.