2016

Use of Bridging Strategy between the Ensemble Kalman Filter and Particle Filter for the Measurements with Various Quasi-Gaussian Noise

Sumathi Prabhakaran Jeyakumari
Minnesota State University Mankato

Recommended Citation

Prabhakaran Jeyakumari, S. (2016). Use of Bridging Strategy between the Ensemble Kalman Filter and Particle Filter for the Measurements with Various Quasi-Gaussian Noise [Master's thesis, Minnesota State University, Mankato]. Cornerstone: A Collection of Scholarly and Creative Works for Minnesota State University, Mankato. https://cornerstone.lib.mnsu.edu/etds/640/

Use of Bridging Strategy between the Ensemble Kalman Filter and Particle Filter for the Measurements with Various Quasi-Gaussian Noise

By

Sumathi PrabhakaranJeyakumari

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science Degree

In

Electrical Engineering

Minnesota State University, Mankato

Mankato, Minnesota

May 2016

Use of Bridging Strategy between the Ensemble Kalman Filter and Particle Filter for the Measurements with Various Quasi-Gaussian Noise

Sumathi PrabhakaranJeyakumari

Date:- 05/05/2016

This thesis has been examined and approved by the following members of the Student's committee.

Dr.Vincent Winstead – Advisor **(**ECET)

Dr. Han-Way Huang- Committee Member (ECET)

Dr. Namyong Lee - Committee Member (Math)

Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Vincent Winstead for his continuous support during my master's thesis, including research as well as the coursework in Electrical Engineering at Minnesota State University. I would like to thank him for his guidance, encouragement, patience, motivation and enthusiasm. He gave me enough freedom and allowed this thesis work to be my own, but he also steered me in the right direction whenever I thought I could not move forward. This accomplishment would not have been possible without him.

Besides my advisor, I am also very grateful for the rest of my thesis committee,
Prof. Han-Way Huang and Prof. Namyong Lee, for their encouragement, insightful comments, and questions. I thank Prof. Namyong Lee gave moral support and taught me to add passion to my work and also learned MATLAB^{TM} simulation in his course Intro to Modeling and Simulation which is a major part of my thesis work.

In addition my sincere thanks to Prof. Julio Mandojana for teaching me Advanced Control System, introducing nonlinear system analysis and discussed optimization techniques which helped me to work on my thesis. I thank Dr. Muhammad Khaliq for inspiring me in time management.

Finally, I must express my very profound gratitude to my family and to my parents for providing me with continuous support and encouragement throughout my years of study and through the process of researching and writing this thesis. Thank you.

This thesis is dedicated to my loving husband and daughter

TABLE OF CONTENTS

List of Tables

List of Figures

Abstract

Filtering and estimation are two important tools of engineering. Whenever the state of the system needs to be estimated from the noisy sensor measurements, some kind of state estimator is used. If the dynamics of the system and observation model are linear under Gaussian conditions, the root mean squared error can be computed using the Kalman Filter. But practically, noise frequently enters the system as not strictly Gaussian. Therefore, the Kalman Filter does not necessarily provide the better estimate. Hence the estimation of the nonlinear system under non-Gaussian or quasi-Gaussian noise is of an acute interest. There are many versions of the Kalman filter such as the Extended Kalman filter, the Unscented Kalman filter, the Ensemble Kalman filter, the Particle filter, etc., each having their own disadvantages. In this thesis work I used a bridging strategy between the Ensemble Kalman filter and Particle filter called an Ensemble Kalman Particle filter. This filter works well in nonlinear system and non-Gaussian measurements as well. I analyzed this filter using MATLAB^{TM} simulation and also applied Gaussian Noise, non-zero mean Gaussian Noise, quasi-Gaussian noise (with drift), random walk and Laplacian Noise. I applied these noises and compared the performances of the Particle filter and the Ensemble Kalman Particle filter in the presence of linear and nonlinear observations which leads to the conclusion that the Ensemble Kalman Particle filter yields the minimum error estimate. I also found the optimum value for the tuning parameter which is used to bridge the two filters using Monte Carlo Simulation.

Chapter 1

Introduction

Estimation theory is mostly used in electronic signal processing systems like radar, sonar, image analysis, biomedicine, communications, controls, seismology, speech etc. and also includes diverse areas of science such as meteorology, biostatistics, econometrics, and geology as well as many others. Problems such as target tracking, time series analysis, communication and satellite navigation require the need to estimate values of the state or a group of parameters. For example, in radar and sonar the desired parameter is the position of the aircraft and the position of a target such as a submarine respectively. Estimation of the state of a stochastic system from a noisy measurements or uncertainties is a main problem faced in the almost all fields. The main objective is to estimate the signal from the noisy measurements. These problems come under the classification of Bayesian Inference problems, a sub category of statistical inference where the likelihood of a hypothesis is updated sequentially in the presence of observed data. These kinds of problems assume that the present state of the system depends only on the state at the prior instant. The state and their mathematical relations like mean, variance etc. may be known or may be hypothesized based on experience. In both cases the state space model connecting the observation and the states of interest is a probabilistic one, because of the presence of uncertainties and noises. Hence one needs a best estimation approach for the optimum solution.

1.1 The Optimum Solution

The optimum solution of a model which is entirely linear and the noises/uncertainties involved are Gaussian distributions with known parameters, is given by the Kalman filter [19]. The discrete Kalman filter is very useful and has been found to solve a wide variety of problems encountered in different fields of science and technology as discussed earlier. Kalman filtering can provide minimum estimation errors. Also the performance of Kalman filtering is easy to verify and it is easy to implement. The linear and Gaussian distribution model represents a small part of the Bayesian Inference Problems. In most cases the dynamics of the system will not be linear with Gaussian noise but nonlinear with non-Gaussian noice. Also higher dimensional systems add complexity to the problems. In this regard Kalman filter cannot provide an optimum solution.

1.2 The Sub-optimal solutions

To solve problem of the nonlinearity involves linearizing the nonlinear system process and measurement functions. Different Kalman filters linearize the system function in different ways. Among these various filters the Extended Kalman Filter (ExKF) can be used when the problem involves a nonlinear function. ExKF linearizes the nonlinear function locally by using the first term of its Taylor series expansion. If the complexity of the system is large, ExKF uses higher order terms of the expansion. In that case the linearization errors are not negligible which eventually leads to an inconsistent estimate, and also increased computational complexity. In this filter it assumes the noise has a Gaussian distribution and uses the Kalman Filter equations to obtain the estimate at each step of estimation. The major drawback of this approach is that it assumes the noise is Gaussian so ExKF does not give an optimum solution for the non-Gaussian distribution. The detailed algorithm is provided in chapter 3.

Another approach is the approximate grid based methods in which discretizing a continuous state space to get an optimal solution. In this method the continuous state domain is divided into a finite number of states and probability density functions of the estimations are modified into probability mass functions. The prediction step and update step equations are computed using the conditional probability of the state with respect to observation. But the problem with this method is the original space should be known and the area of high occurrence should be known. In most of the real time problems this is not possible. Another drawback is the truncation error originating from certain part of the state space.

1.3 Monte Carlo Methods

The Ensemble Kalman Filter and the particle filters are based on the Monte Carlo estimation methods. First of all a region of possible input points should be defined, which is equivalent to define the priori probability distribution in the Bayesian Estimation problem. Then a fixed number of samples are generated from this distribution. With the use of these sample points, estimation of the parameters are performed. Therefore these methods depend on the law of large numbers so they replace integration involving probability terms with computed sums and averages. The advantage of the Monte Carlo method is that it is easy to solve systems having complicated integrals as mentioned earlier. The Ensemble Kalman filter and the Particle filter is used extensively in the Bayesian Inference problems especially in case of high dimensional non-Gaussian and nonlinear models.

The Particle filter [20] is a recursive filter which generates the variables from a sample population, collaborates a specific weight and then computes a weighted average to find the final estimate. Even if the states are unknown it will draw from the approximate distribution of the samples. As the sample size increases, the characterization of the Monte Carlo method leads to the true value and also the filter tends to the optimal Bayesian estimate.

The Ensemble Kalman Filter [14] is an extension of the discrete Kalman filter (this filter uses the Monte Carlo methods to generate the sample of model states) which is used for the nonlinear Bayesian filtering. Using the samples, states, and observation it will estimate the final result. First of all, predict the forecast ensemble estimates based on the estimates at the prior instant. Then the forecast ensembles are adjusted using the ensemble Kalman gain along with the most recent observation. The Ensemble Kalman filter will provide the best estimate for the Gaussian distribution. It has been shown that these filters work well and optimal solutions will converge with the Kalman filter solution under linear and Gaussian distribution [21 , 22].

The Ensemble Kalman filter (EnKF) and the Particle filter (PF) methods approximate the probability density function in different ways. The EnKF only approximates the mean and covariance of the state through a series of equally weighted ensemble members. The analysis of EnKF, which is a weighted combination of the prediction and observation through Kalman gain, and also it updates each ensemble member based on its distance from the observation in the state space. But in the case of particle filter only updates the weight of each particle in the analysis step without updating the particle itself. Because most particles may have small weights, a large number of particles are required to prevent filter degeneracy, making the particle filter impractical for high-dimensional models.

The Ensemble Kalman Particle filter (EnKPF) takes the advantages of both ensemble Kalman filter and the Particle filter. This approach uses a combined analysis scheme including both the EnKF and the Particle Filter by using a controllable index (i.e., tuning) parameter. This new analysis scheme of the EnKPF is not only updates the particles but also considers its weights. Therefore this thesis analyzes the Ensemble Kalman Particle filter and applied the non-Gaussian noise.

1.4 Problem Description

As many practical problems of Bayesian inference are nonlinear with non-Gaussian distribution, this work assumes the observations are related nonlinearly to the states and also distributed under the non-Gaussian support. For this type of model we compare the root mean squared error of the Ensemble Kalman Filter, Particle filter and the new method of Ensemble Kalman Particle filter. It will be shown that the Ensemble Kalman Particle filter yields minimum error.

The problem being considered is to determine or predict a parameter x at a given time step and a noisy observation y. The variable x is assumed to follow a known probabilistic model that depends on the prior state. The observed variable y is a function of x under the non-Gaussian noise. We then consider various filtering approaches applied to both Gaussian and non-Gaussian noises and then analyzed the performance by changing the tuning parameter gamma. The EnKPF is demonstrated using MATLAB^{TM} simulation. Then we briefly discuss the simulation results and the future work in this regard. In order to understand this filter it is required to first understand the basic principle of the classical discrete Kalman filter which yields the best estimate for the Bayesian estimation problem.

Chapter 2

Background

2.1 Introduction to state estimation

Estimation can apply to both static and dynamic systems. A dynamic system is a system in which the output at any particular time is completely determined by input samples at the same time as well as at other times including the past or the future. Since the output depends on past or future input samples, this system needs a memory to store such samples. Hence, a dynamic system has a memory. In this system there are system states and the evolution of the system state over time. The system state at time t is an instantaneous description of the system which means it is sufficient to predict the future states of the system without considering the prior states. The evolution of the system state indicates sequence or continuous trajectory through the space of all possible system states. This means that these types of states hold all the information about the past. Therefore there is a need to know the states at a given point of time and the inputs from there on (it may contain all the information of system from the beginning), which may require an infinite number of parameters to describe the system. This is of course impossible. Therefore there is a need to model all the space of possible system states. This is called the state space of the dynamical system. With the use of a state space model the past is encoded in a finite number of terms, which is the advantage of this modeling and also the state space model allows the description of the state, inputs, and the output.
A discrete state space model is of the following form,

$$X(k+1) = AX(k) + BU(k) \tag{2.1}$$
$$Y(k) = CX(k) + DU(k) \tag{2.2}$$

Where $X(.) \in R^n$ is the state vector and n is the number of states, A is the system matrix, B is the input matrix, $U(.) \in R^m$ is the input vector and m is the number of inputs, and k is the discrete time index. Matrices A and B are of appropriate dimension. The future state can be expressed in terms of input and past state, so it has a finite description. Where Y (.) is the output vector, C is the output matrix and D is the feed through / feed forward matrix. The output which is composed of system measurements are actually related to the state and the input. Matrices C and D are also of appropriate dimensions.

The outputs are measured but states may not be measured directly. Some states may affect the dynamics of the system, which may also affect the output of the system but may not be measured. This problem actually demands estimation, if the states need to be determined.

Although not always necessary state estimation may still be useful. Such state estimates can be used to construct a control mechanism, such as a rocket or a missile, which is supposed to hit another missile or an aircraft. The reference signal that usually comes in a control system defines the desired missile trajectory. This reference signal has a mixture of two properties. First of all it must follow the reference faithfully, and then it must reject all disturbances. This is the desired property of all tracking control systems. However if the problem is to compute the reference itself this problem is known as guidance. To intercept a moving object all its motion parameters such as its present position, its velocity, its acceleration, etc., are required. These are the states of the previously mentioned system. The position is measured with the use of sensors and then velocity and acceleration are estimated. But a non-trivial problem occurs when one tries to estimate the velocity and acceleration in the absence of a measurement of the position. In such a situation one needs to consider the various estimation methods such as a state observer via an estimation using Kalman filtering and an extended estimation.

2.2 State space observation

State observation is used to estimate the parameters for ideal conditions. Linear systems lend themselves to a simple model representation which can be used for state observation such as

$$X(k+1) = AX(k) + BU(k) \tag{2.3}$$

$$Y(K) = CX(k) \tag{2.4}$$

In this approach the input should be known without any disturbance (which is the ideal condition), initial conditions of the state are unknown or assumed zero and also there are no measurement noises. Hence a state observer is used to perfectly recreate the state dynamics. This situation is not realistic.

Now consider that there is some noise in the measurement and also assume some uncertainty in the inputs, which implies some input may be known but some other inputs are unknown. The other kinds of inputs are called disturbance inputs, which are applied by the environment. Most disturbances are immeasurable in real time. A common way to characterize the unknown or uncertain things is to have a probabilistic distribution. Bayesian state estimation is used to solve these kinds of problems.

2.3 Bayesian state estimation

Bayesian theory is used to solve these kinds of problems. Bayesian theory was originally discovered by the British researcher Thomas Bayes in a posthumous publication in 1763 [1]. The renowned Bayes theorem specifies the fundamental probability law determining the process of logical inference. Bayesian theory (e.g., [2]) is a branch of mathematical probability theory that allows people to model the uncertainty about the world and the outcomes of interest by incorporating prior knowledge and observations. In Bayesian analysis, the probability is described as a conditional measure of uncertainty, is one of the popular methods used to solve the estimation problems [3]. Let's introduce some fundamental Bayesian statistics. The state estimation problem has two components. The first part is associated with an accurate prediction of the sensor measurements of the state. For example a prediction of the position similar to the measured quantities of a position sensor which in turn yields a prior estimate of state. The second part is associated with joining the measured observation with the predicted one to form a posterior estimate of the state. (The equations given below are obtained from [4]).

Consider two random variables, $\boldsymbol{x}$ and $\boldsymbol{y}$**.** This x can take one of $\boldsymbol{N_X}$ values and y can take one of $\boldsymbol{N_Y}$ values, that is $\boldsymbol{x \in N_X, y \in N_Y}$.

The joint probability of observing when $\boldsymbol{x = x_i}$ and $\boldsymbol{y = y_j}$ is $\boldsymbol{Pr(x=x_i, y=y_i)}$

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are statistically independent, this joint probability is just the product of the probabilities

$$\boldsymbol{Pr(x=x_i)\ Pr(y=y_i)} \qquad (2.5)$$

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are statistically dependent, take the probability that $\boldsymbol{x = x_i}$ conditioned on the event $y=y_j$.

This is the conditional probability $\boldsymbol{Pr(x=x_i|\ y=y_j)}$

Bayes's theorem is a result of the property that joint probability is commutative, i.e.

$$\boldsymbol{Pr(x=x_i, y=y_j) = Pr(y=y_j\ , x=x_i\ )} \qquad (2.6)$$

Expanding both sides of with the equation for conditional probability will yield [3]

$$\boldsymbol{Pr(x=x_i|\ y=y_j)\ Pr(y=y_i) = Pr(y=y_j|\ x=x_i)\ Pr(x=x_i)} \qquad (2.7)$$

Bayes theorem is then obtained by [3]

$$Pr(x=x_i \mid y=y_j) = Pr(y=y_j \mid x=x_i)Pr(x=x_i)/Pr(y=y_i) \tag{2.8}$$

This theorem is applicable not only for the probability values but also for the probability distributions.

So if x and y are continuous random variables then according to Bayes theorem the relation between the distribution is

$$P(x|y)= P(y|x)\,\frac{P(x)}{P(y)} \tag{2.9}$$

If the variable x represents the state of the dynamical system and y represents an observation (or measured output from a sensor), ***P(y|x)*** is the likelihood of an observation given that the state represents the true underlying model. ***P(x|y)*** yields the probability that the model is correct "after the fact" given a collected an observation. Therefore it is called the posterior distribution [4].

P(x) is the probability distribution of the state independent of the observation or the prior ***x. P(y***) is the prior probability of observation.

There are three types of intractable problems inherently related to the Bayesian statistics: [3]

1. Normalization: To find the posterior ***P(x|y)*** with the given prior ***P(x***) and the likelihood ***P(y|x)*** [3]

$$P(x|y\,) = \frac{P(y|x)P(x)}{\int_X P(y|x)P(x)dx} \tag{2.10}$$

2. Marginalization: Given the joint posterior *(x,z)*,the marginal posterior[3]

$$P(x|y\,)=\int_z P(x,z|y)dz \tag{2.11}$$

3. Expectation for a given conditional probability distribution function [3]

$$E_{P(x|y\,)}[f(x)]=\int f(x)\,P(x|y\,)dx. \tag{2.12}$$

where ***f(x)*** is the system function

There are three major steps in Bayesian Analysis

1) Select the model with given data and assumed priors.
2) Estimate the parameters to fit the data and assumed priors.
3) Update the parameters of the prior.

Now consider the situation such that the observation changes with respect to time. Recursive Bayesian estimation is a method to estimate the real value of an observed variable that changes with time.

2.4 Recursive Bayesian Estimation

In this method there are two main assumptions. First of all the states follow the first order Markov process. This process is a random process that undergoes transitions from one state to another on a state space. But the probability distribution of the future state depends only on the current state from [3]

$$P\,(X_k|X_0, X1, X2, X3, \ldots, X_{k-1}) = P(X_k|X_{k-1}) \qquad (2.13)$$

Second assumption is the observations are independent of the given states.

From Bayes rule the conditional probability [3]

$$P(X_k|Y_k) = P(Y_k|X_k)\,\frac{P(x)}{P(y)} \qquad (2.14)$$

$$P(X_k\,|Y_k) = \frac{P(y_k, y_{k-1}|x_k)P(\,x_k)}{P(y_k, y_{k-1})}$$

$$= \frac{P(y_k, y_{k-1}|x_k)P(x_k)P(y_k, y_{k-1}|x_k)\;P(\,y_{k-1}\,|Xk)\;P(x_k)\;)}{P(y_k|y_{k-1})\;Py_{k-1})}$$

$$= \frac{P(y_k, y_{k-1}|x_k)\;P(x_k|y_{k-1})\;P(y_{k-1}\,)\;P(x_k)}{P(y_k|y_{k-1})\;P(y_{k-1})\;P(x_k)}$$

$$= \frac{P(y_k|x_k)P(x_k|y_{k-1})}{P(y_k|y_{k-1})} \qquad (2.15)$$

Here $\boldsymbol{P(X_k|Y_{k-1})}$ is the prior which defines the knowledge of the model [3]

$$\boldsymbol{P(X_k|Y_{k-1})=\int P(x_k|x_{k-1})P(x_{k-1}|y_{k-1})dx_k = 1} \quad , \qquad (2.16)$$

and $\boldsymbol{P(X_k|Y_{k-1})}$ is the transition density of the state.

Chapter 3

Kalman Filter

3.1 Discrete Kalman Filter

The Kalman Filter is the special case in the recursive Bayesian state estimation. Now consider the measurement error (practically the sensors have measurement error). Even though it is a linear system, in these model equations the system matrix, input, and output matrices are time varying since the Kalman filter formulation is done for the time varying case. Let's assume that the system is time invariant in order to reduce the complexity of estimation [5].

The system model where k is the time step

$$X(k+1) = A\,X(k) + B\,U(k) + w \qquad (3.1)$$

$$Y(k) = C\,X(k) + D\,U(k) + v \qquad (3.2)$$

Where $\boldsymbol{w} \in \mathrm{R}^n$ is the system noise which is applied through the input like disturbances unknown noise. And $\boldsymbol{v} \in \mathrm{R}^m$ is the measurement noise term.

The assumptions for the Kalman filter are as follows [5]

1) $E(w)$ and $E(v)$is zero for all $\boldsymbol{k}$.(E() is the expectation)
2) The correlation and covariance are same because the mean is zero.
3) $E(w\,w^T) = Q$ is called process noise covariance and it should be positive definite.
4) $E(v\,v^T) = R$
5) The input noise and the observation noise are mutually uncorrelated that is $E(w\ v^T) = E(X_0\,w^T) = E(X_0\,v^T) = 0$ for all $\boldsymbol{k}$.

$\hat{x}_k^-$ is the a priori state estimate at step $\boldsymbol{k}$ given knowledge of the process prior to step $\boldsymbol{k}$.

$\hat{x}_k$ is the a posteriori state estimate at step $\boldsymbol{k}$ given measurement y_k

The priori and a posteriori estimate errors are as follows [5]

$$e_k^- = x_k - \hat{x}_k^- \qquad (3.3)$$

$$e_k = x_k - \hat{x}_k \qquad (3.4)$$

The priori estimate error covariance is $P_k^- = E[e_k^- e_k^{-T}]$. The posteriori estimate error covariance is $P_k = E[e_k e_k^T]$. The a posteriori estimate is computed as

$$\hat{x}_k = \hat{x}_k^- + K(y_k - C\hat{x}_k^-) \qquad (3.5)$$

where $(y_k - C\hat{x}_k^-)$ is called the measurement innovation or the residual and

$\boldsymbol{K}$ is the Kalman gain which minimizes the posteriori error covariance.

In order to minimize the error,

$e_k = x_k - \hat{x}_k$, substitute equation (3.5) in the error

$$\hat{x}_k = \hat{x}_k^- + K(y_k - C\hat{x}_k^-) \quad (3.6)$$

yielding gain **K** at each step

$$K_k = P_k^- C^T (C P_k^- C^T + R)^{-1} \quad (3.7)$$

When the measurement error covariance R approaches to zero

$K_k = C^{-1}$ so the gain weights the residual more heavily.

When the a priori estimate error covariance P_k^- approaches to zero, the gain approaches to zero i.e. $K_k = 0$. The gain weights the residual less heavily. Which means when R tends to zero the actual measurement y_k is accurate compare with the predicted measurement $C\hat{x}_k^-$. When the priori estimate error covariance P_k^- tends to zero then the predicted measurement $C\hat{x}_k^-$ is accurate than the actual measurement. Probability of a priori estimate $\hat{x}_k^-$ conditioned on all prior measurements y_k (Bayes Rule) yields

$E[X_k] = \hat{x}_k$ (this is the mean of the state distribution)

$E[(x_k - \hat{x}_k^-)(x_k - \hat{x}_k^-)^T] = P_k$ (variance of the distribution.)

3.2 The Kalman filter Algorithm

The Kalman filter equations are categorized into two forms. First one is to predict (time update) which is used to obtain the a priori estimates for the next time step. The other one is the measurement update equations are used for the feedback that is incorporating the new measurement into the predicted estimate to get the a posteriori estimate [5].

The predictor equations

$\hat{x}_k^- = A\,\hat{x}_{k-1} + B\,u_k$

$P_k^- = A\,P_{k-1}A^T + Q$

update equations

$K_k = P_k^- C^T (C P_k^- C^T + R)^{-1}$

$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$

$P_k = (I - K_k C)P_k^-$ (3.8)

After each prediction and updating step, the process is repeated with the use of previous a posteriori estimates used to predict the new a priori estimates. This recursive nature makes the Kalman filter practical. Kalman filters yields optimal solution for the linear systems given process noise and the measurement noise have zero mean with Gaussian support. Now consider the system is nonlinear but all other assumptions are the same including Gaussian noise. In that case linearization technique are used in the approximation, i.e. the Extended Kalman Filter.

(The complete derivation of the Kalman filter equations can be found at "Kalman Filtering Theory and Practice using MATLAB" by Mohinder S.Grewal, Angus P. Andrews. In chapter 4, section 4.2 Kalman Filter)

3.3 Extended Kalman Filter

Extended Kalman Filtering provides the estimate by linearizing the nonlinear model of the system. That is, it linearizes about the current mean and covariance. Taylor series is used to linearize the model using the partial derivatives of the process and measurement functions to provide the estimate.

Now assume the process is modeled by a nonlinear differential equation.

$$X_k = f(x_{k-1}, u_k, w_{k-1}) \text{ , where } x \in R^n \tag{3.9}$$

With measurement $y \in R^m$,

$$Y_k = h(X_k, v_k) \tag{3.10}$$

The main difference from the Kalman filter algorithm is that one linearizes the process and measurement functions using a Taylor series. Hence, the system matrix and output matrix of the state space model are replaced with Jacobian matrixes by using the partial derivatives.

Using new equations to linearize an estimate about Kalman Filter equation yields

$$X_k = \tilde{x}_k + A\,(x_{k-1} - \hat{x}_{k-1}) + Ww_{k-1} \tag{3.11}$$

$$Y_k = \tilde{y}k + H(x_k - \tilde{x}_k) + Vv_k \tag{3.12}$$

where X_k and Y_k are the true state and measurement vectors,

$\tilde{x}_k$ and $\tilde{y}_k$ are the approximate state and measurement vectors,

$\hat{x}_k$ is a posterior estimate of the state at step ***k***, and

w_{k-1} and v_k are the process and measurement noise.

For simplicity did not use the time step ***k*** for the Jacobians even if they change with respect to time.

The matrix ***A*** is the Jacobian matrix of the partial derivatives of ***f*** with respect to ***x***. That is

$$A_{[i,j]} = \frac{\partial f[i]}{\partial x[j]}\,(\hat{x}_{k-1}\ , u_k, 0) \tag{3.13}$$

where i, j is the number of state equations and the number of states respectively. u_k is the input.

W is the Jacobian matrix of ***f*** with respect to ***w***

$$W_{[i,j]}=\frac{\partial \boldsymbol{f}[\boldsymbol{i}]}{\partial \boldsymbol{w}[\boldsymbol{j}]}(\hat{x}_{k-1},u_k,0) \qquad (3.14)$$

H is the Jacobian matrix of ***h*** with respect to ***x***

$$H_{[i,j]}=\frac{\partial \boldsymbol{h}[\boldsymbol{i}]}{\partial \boldsymbol{x}[\boldsymbol{j}]}(\tilde{x},0) \qquad (3.15)$$

V is the Jacobian matrix ***h*** with respect to ***v***

$$V_{[i,j]}=\frac{\partial \boldsymbol{h}[\boldsymbol{i}]}{\partial \boldsymbol{v}[\boldsymbol{j}]}(\tilde{x},0) \qquad (3.16)$$

3.4 Extended Kalman Filter Algorithm

Prediction equations [5]

$\hat{x}_k^- = f(\hat{x}_{k-1},u_k,0)$

$P_k^- = A_k P_{k-1} A_k^T + W_k Q_{k-1} W_k^T$

update equations [5]

$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + V_k R_k V_k^T)^{-1}$

$\hat{x}_k = \hat{x}_k^- + K_k(y_k - h(\hat{x}_k^-,0))$

$P_k = (I - K_k H_k) P_k^-$ (3.17)

Extended Kalman Filter is an improvised Kalman filter and simply an ad-hoc state estimator. Even though this filter is used for the nonlinear estimation it has some disadvantage. Since it uses Taylor series to linearize it can cause large truncation errors. Extended Kalman filtering gives the best optimal solution for simple nonlinear models. There is no one to one mapping between the measurements $\boldsymbol{Y_k}$ and the state via $\boldsymbol{h}$. $\boldsymbol{H_k}$ affects the Kalman Gain which only magnifies the residual but does not affect the state. Therefore it has a chance to diverge. Hence, for a highly nonlinear system one can apply a new approximation method called Unscented Kalman Filter.

3.5 Unscented Kalman Filter

The Unscented Kalman filter is also called Sigma-Point Kalman Filter or Linear regression Kalman Filter which uses the statistical linearization technique [7, 8]. This method is used to linearize a nonlinear function of the states through a linear regression between n points drawn from the prior distribution of the states. This linearization technique is an improvement over Taylor series linearization [9].

The state distribution of the Extended Kalman filter is analytically propagated through the first order linearization of the nonlinear system, in which there is a chance of divergence in the posterior mean and covariance. The Unscented Kalman Filter overcomes this problem through a deterministic sampling method called unscented transformation [10]. Consider the same nonlinear system as in the Extended Kaman Filter

$X_k = f(x_{k-1}, u_k, w_{k-1})$, where $x \in R^n$

$Y_k = h(X_k, v_k)$, with measurement $y \in R^m$.

Now assume the initial state x_0 is a random vector with known mean $\bar{x}_0 = E(x_0)$. The covariance $P_0 = E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T]$ where $\bar{x}_k = E(x_k)$ is the mean of the state vector and $P_k = E[(x_k - \bar{x}_k)(x_k - \bar{x}_k)^T]$ is the covariance.

In the Unscented Transform,

1) Selection of Sigma Points (the sigma points are denoting by χ). The number of sigma points is $2n+1$ where n is the dimension of the state [11]

   $\chi^{[0]} = \bar{x}_k$ first sigma point is the mean. [i] = 0 *(3.18)*

   $\chi^{[i]} = \bar{x}_k + (\sqrt{(n+\lambda)Pk})_i$ for [i]=1,2,……n *(3.19)*

   $\chi^{[i]} = \bar{x}_k - (\sqrt{(n+\lambda)Pk})_{i-n}$ for [i]=n+1,…………..2n *(3.20)*

   Note: $\sqrt{(n+\lambda)Pk}$ - is a matrix square root there is two methods to compute the square root, one is diagonalization and another method is Cholesky Matrix Square root.

2) Set the weights (the weights are denoted by ω) [12]

   $\sum_i \omega^{[i]} = 1$ *(3.21)*

   $\bar{x} = \sum_i \omega^{[i]} \chi^{[i]}$ *(3.22)*

   $P_k = \sum_i \omega^{[i]} (\chi^{[i]} - \bar{x}_k)(\chi^{[i]} - \bar{x}_k)^T$ *(3.23)*

   There is no unique solution to compute the sigma points and the weights.

3) Weighted sample mean and covariance [11,12]

   $\omega_m^{[0]} = \frac{\lambda}{n+\lambda}$ *(3.24)*

   $\omega_c^{[0]} = \omega_m^{[0]} + (1-\alpha^2+\beta)$ where λ, α, β are the scaling parameters *(3.25)*

   $\omega_c^{[i]} = \omega_m^{[i]} = \frac{1}{2(n+\lambda)}$ for i= 1,2,. . . 2n (n- is the number of states) *(3.26)*

   The parameter λ can be computed by $\lambda = \alpha^2(n+\kappa)-n$

   Where $\omega_m^{[0]}$, $\omega_m^{[i]}$ are needed in order to compute the mean and

$\omega_c^{[0]}$,$\omega_c^{[i]}$ are needed to compute the variance.

The values for α∈(0,1], β=2, κ≥0and also these values depend

on how far the sigma points are from the mean [11, 12].

4) Transform these points through a nonlinear mapping

$\boldsymbol{Y_k} = \boldsymbol{G(x_k,w)}$

where $\boldsymbol{x_k}$ is the input, $\boldsymbol{Y_k}$ is the output, $\boldsymbol{G(.)}$ is the nonlinear map parameterized by the vector $\boldsymbol{w}$ weights

5) Compute the mean and covariance from weighted transformed points

$$\boldsymbol{\bar{x}'}=\sum_{i=0}^{2n} \omega_m^{[i]} \boldsymbol{G}(\boldsymbol{\chi}[\mathrm{i}]) \qquad (3.27)$$

$$\boldsymbol{Pk'} = \sum_{i=0}^{2n} \omega_m^{[i]} \boldsymbol{G}(\boldsymbol{\chi}[\mathrm{i}] - \boldsymbol{\bar{x}'})\, \boldsymbol{G}(\boldsymbol{\chi}[\mathrm{i}] - \boldsymbol{\bar{x}'})^{\mathrm{T}} \qquad (3.28)$$

3.6 Unscented Kalman Filter Algorithm

Prediction [13]

$$\boldsymbol{\hat{x}_k}^- = \sum_{i=0}^{2n} \omega_m^{[i]} \boldsymbol{\chi}_{\mathbf{i(k|k-1)}}$$

$$\boldsymbol{P_k}^- = \sum_{i=0}^{2n} \boldsymbol{\omega_c^{[i]}} (\boldsymbol{\chi}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{x}_k}^-)(\boldsymbol{\chi}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{x}_k}^-)^{\mathrm{T}} + \boldsymbol{Q}$$

$$\boldsymbol{Y_{k|k-1}} = \boldsymbol{H}(\boldsymbol{\chi}_{\mathbf{(k|k-1)}})$$

$$\boldsymbol{\hat{y}_k}^- = \sum_{i=0}^{2n} \boldsymbol{\omega_m^{[i]}} \boldsymbol{Y_{k|k-1}} \qquad (3.29)$$

Update [13]

$$\boldsymbol{P\hat{y}_k\text{-}\hat{y}_k\text{-}} = \sum_{i=0}^{2n} \boldsymbol{\omega_c^{[i]}} (\mathbf{Y}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{y}_k}^-)(\mathbf{Y}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{y}_k}^-)^{\mathrm{T}} + \boldsymbol{R}$$

$$\boldsymbol{P\hat{x}_k\text{-}\hat{y}_k\text{-}} = \sum_{i=0}^{2n} \boldsymbol{\omega_c^{[i]}} (\boldsymbol{\chi}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{x}_k}^-)(\mathbf{Y}_{\mathbf{i(k|k-1)}} - \boldsymbol{\hat{y}_k}^-)^{\mathrm{T}}$$

$$\boldsymbol{K_k} = \boldsymbol{P\hat{x}_k\text{-}\hat{y}_k\text{-}} \,(\boldsymbol{P\hat{y}_k\text{-}\hat{y}_k\text{-}})^{-1}$$

$$\boldsymbol{\hat{x}_k} = \boldsymbol{\hat{x}_k}^- + \boldsymbol{K_k}(\boldsymbol{y_k} - \boldsymbol{\hat{y}_k}^-)$$

$$\boldsymbol{P_k} = \boldsymbol{P_k}^- - \boldsymbol{K_k P\hat{y}_k\text{-}\hat{y}_k\text{-}K_k}^{T} \qquad (3.30)$$

Where Q and R are process and measurement noises covariance respectively.

Even though Unscented estimation is accurate, this method is based on a small set of sigma points therefore it is not a truly global approximation technique. Computation cost is greater because of the Cholesky factorization on every step. This yields slower computation when compared with other techniques. And it is also applied to models driven by Gaussian noises. For these reasons this filter is difficult to implement.

Another method which can be applied to the nonlinear state estimation is known as the Ensemble Kalman filter. This filter can be used in the models of extremely higher order and nonlinear, if the initial states are uncertain and have a large number of measurements.

Chapter 4

Ensemble Kalman filter

This filter is derived from the basic Kalman filter which provides the best estimation in linear Gaussian models. Also it provides sub optimal solutions for extremely high order and nonlinear systems. This filter obtains an estimate by using the first and second moments of the error terms. Basically this filter is consists of Monte Carlo approximations of the Kalman filter in which the actual covariance is replaced by the ensembles covariance.

4.1 Gaussian Linear observation

Much of the material summarized in this chapter is derived from Shen & Tang [17]. Consider an ensemble of state estimates, which holds the initial probability distribution of the state. In order to capture statistical information of the predicted states, the sample points are propagated to the true nonlinear system and the probability density function of the actual state covariance of the prediction error is approximated by the ensemble of the estimates [14]. In unscented Kalman filtering the number of sample points is selected deterministically from a minimal set of points. But in Ensemble Kalman filter the number of ensembles can vary and it is also assumed that the prediction distribution is Gaussian. The application of Ensemble Kalman Filtering is given [14].

The prior ensembles are denoted by $X^f \in R^{LxN}$ where L is the number of states and N is the ensemble size.

$\boldsymbol{X^f = (X^f_1, X^f_2, \ldots\ldots X^f_i, \ldots X^f_N)}$ these ensembles are returned as updated ensembles [14].

The parameters f and i denote the ith forecast ensemble member.

The emperical mean and covariance are defined by

$$\overline{\boldsymbol{X^f}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{X_i^f} \qquad (4.1)$$

$$\overline{\boldsymbol{P^f}} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{X_i^f} - \overline{\boldsymbol{X^f}})(\boldsymbol{X_i^f} - \overline{\boldsymbol{X^f}})^T \qquad (4.2)$$

The Ensemble Kalman Filter performs the Kalman filter formula for each ensemble member, i.e.

$$\boldsymbol{X_i^a} = \boldsymbol{X_i^f} + K(y_i - h(X_i^f) \qquad (4.3)$$

where K is the kalman gain and X_i^a is the updated ensembles for the linear measurement function

For i= 1,2,. . . . .,N.

The measurement is

$$\boldsymbol{y_i}= \boldsymbol{y}+\boldsymbol{v_i} \tag{4.4}$$

where v_i is a random variable with zero mean and covariance ***R.***

If the measurement function is linear under the additive noise then $\boldsymbol{y_k}=\boldsymbol{Hx_k}+\boldsymbol{\zeta}$.

The Kalman gain is defined by

$$\boldsymbol{K}=\overline{\boldsymbol{P^f}}\boldsymbol{H^T}(\boldsymbol{H}\overline{\boldsymbol{P^f}}\boldsymbol{H^T}+\boldsymbol{R})^{-1} \ . \tag{4.5}$$

4.2 Non Gaussian and nonlinear observation

For the nonlinear measurement function $\overline{P^f}H^T$ and H$\overline{P^f}H^T$ can be calculted as [14]

$$\overline{\boldsymbol{P^f}}\boldsymbol{H^T}=\frac{\boldsymbol{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{X_i^f}-\overline{\boldsymbol{X^f}})[\boldsymbol{h}(\boldsymbol{X_i^f})-\overline{\boldsymbol{h}(\boldsymbol{X^f})}]^{\boldsymbol{T}} \tag{4.6}$$

$$\boldsymbol{H}\overline{\boldsymbol{P^f}}\boldsymbol{H^T} =\frac{\boldsymbol{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{h}(\boldsymbol{X_i^f})-\overline{\boldsymbol{h}(\boldsymbol{X^f})}[\boldsymbol{h}(\boldsymbol{X_i^f})-\overline{\boldsymbol{h}(\boldsymbol{X^f})}]^{\boldsymbol{T}} \tag{4.7}$$

where $\overline{h(X^f)}=\frac{1}{N}\sum_{i=1}^{N}h(X_i^f)$. These equations work well under the following two conditions

1) $\overline{h(X^f)}=h(X_i^f)$
2) Norm($X_i^f-\overline{X^f}$) is small for i= 1,2,. . .N

For a nonlinear model and nonlinear measurement function the ensemble kalman filter gain is

$$\boldsymbol{K}=\boldsymbol{P_{xy}}\boldsymbol{P_{yy}^{-1}} \tag{4.8}$$

where P_{xy} is the cross covariance between the state and observation errors and P_{yy} is the error covariance between the observation and the prediction.

The true value of the state and the observation can be defined as [14]

$$\boldsymbol{X^{tr}}=\boldsymbol{E}(\boldsymbol{X_i^f})+\xi = \overline{\boldsymbol{X^f}}+\xi \tag{4.9}$$

$$\boldsymbol{Y^{tr}}=\boldsymbol{h}(\overline{\boldsymbol{X^f}})+\zeta \tag{4.10}$$

where $\boldsymbol{\xi}$**,** and $\boldsymbol{\zeta}$ are the process noise and the measurement noise. Now

$$\boldsymbol{P_{xy}}=\frac{\boldsymbol{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{X_i^f}-\overline{\boldsymbol{X^f}})[\boldsymbol{h}(\boldsymbol{X_i^f})-\boldsymbol{h}(\overline{\boldsymbol{X^f}})]^{\boldsymbol{T}} \tag{4.11}$$

$$\boldsymbol{P_{yy}}= \frac{\boldsymbol{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}(\boldsymbol{h}(\boldsymbol{X_i^f})-\boldsymbol{h}(\overline{\boldsymbol{X^f}})[\boldsymbol{h}(\boldsymbol{X_i^f})-\boldsymbol{h}(\overline{\boldsymbol{X^f}})]^{\boldsymbol{T}}+ \boldsymbol{R} \tag{4.12}$$

$$X_i^a = X_i^f + K(y_i - h(X_i^f)) \quad (4.13)$$

where X_i^a is the updated state for the nonlinear measurement function given the noise covariance R.

The algorithm is simulated using these equations based on the paper by Shen and Tang [17]. This Ensemble Kalman Filter approach gives better estimates for very small ensemble size. Therefore this filter is suitable for large models. When the ensemble size increases, the performance of the filter also increases but in some extent the performance cannot improve even with the increase in ensemble size. The disadvantage of the ensemble size is the inherent Gaussian assumption.

Chapter 5

The Particle filter

A particle filter [20] based on the Monte Carlo method is used to solve a variety of problems including nonlinearity and high dimensionality. In the literature different terms are used to describe this filter. In this thesis we use [17] for the algorithm. The basic method of this filtering uses a set of particles of probability densities and computes the posterior density function by combining the particles with a set of weights. In some of the methods the same particles are used as trajectories while in the other new particles are generated in each step. In this work we use the same particles. Because this method is similar to Ensemble Kalman filtering so it yields a bridge between both the filters to form the Ensemble Kalman Particle filter (which is considered as the optimum solution).

5.1 Particle filter algorithm

Consider the same nonlinear system as for the previous filter

$$\boldsymbol{X_k = f(x_{k-1}, u_k, w_{k-1})}, \qquad \text{where} \quad \boldsymbol{X_k \in R^n} \tag{5.1}$$

$$\boldsymbol{Y_k = h(X_k, v_k)} \qquad \text{with measurement } \boldsymbol{Y_k \in R^m} \tag{5.2}$$

The purpose of the particle filter is to estimate recursively the posterior distribution of the state $\boldsymbol{P(X_{1:k}|Y_{1:k})}$ or the marginal distribution $\boldsymbol{P(X_k|Y_{1:k})}$ and consequently some functions of the states such as expectation of the sates. The assumption for the particle filter is that probability density function is approximated with weighted particles, that is the likelihood for an ensemble member $\boldsymbol{X_k^{(i)}}$given the observation $\boldsymbol{Y_k}$ to update its weight. The pdf of the analysis at time step ***k-1*** is assumed to be a linear combination of Dirac-Delta functions.

$$\boldsymbol{P(X_{k-1}|Y_{k-1}) = \sum_{i=1}^{N} W_{k-1}^{(i)} \delta\, (X_k - X_{k-1}^{(i)})} \tag{5.3}$$

which is not necessarily Gaussian.

$\boldsymbol{X_{k-1}^{(i)}}$is a particle at time a step ***k-1*** with corresponding weight $\boldsymbol{W_{k-1}^{(i)}}$.

The new particle at step k has marginaldistribution

$$P(X_k|Y_{k-1})=\sum_{i=1}^{N} W_{k-1}^{(i)}\delta\,(X_k - X_k^{(i)}) \qquad (5.4)$$

Based on Bayes' rule the posterior pdf is

$$P(X_k|Y_{k-1})=\frac{1}{A}\sum_{i=1}^{N} P(Y_k|\,X_k^{(i)})\,W_{k-1}^{(i)}\delta\,(X_k - X_k^{(i)}) \qquad (5.5)$$

$$=\sum_{i=1}^{N} W_k^{(i)}\delta\,(X_k - X_k^{(i)}) \text{ where } W_k^{(i)} \propto P(Y_k|\,X_k^{(i)})\,W_{k-1}^{(i)} \qquad (5.6)$$

The estimated pdf of the updated state vectors X_k is

$$\overline{X}_k=\sum_{i=1}^{N} W_k^{(i)}X_k^{(i)}. \qquad (5.7)$$

The mean (the first moment) $\overline{f(x)} = \sum_{i=1}^{N} W_k^{(i)} f(X_k^{(i)})$ where f(x) is a function X_k

The posterior state $f(x)=X^n$ and the likelihood function $P(Y_k|\,X_k^{(i)}) = \Phi(Y_k; h(X_k^{(i)}), R)$

Here observation and the likelihood functions are Gaussian.

$$P(Y|X)= \Phi(y;h(x),R)= A\, exp\{-\frac{1}{2}[y-h(x)]^T R^{-1}[y-h(x)]\} \qquad (5.8)$$

The problem of updating this process is that variance of the weights increases exponentially with respect to time, which means after a few iterations, the distribution of importance weights becomes more and more reduced. This is known as particle degeneracy. In order to measure the degeneracy the effective ensemble size N_{eff} can be computed

$$N_{eff} =1/\ \{\sum_{i=1}^{N}(W^{(i)})^2\} \qquad (5.9)$$

N_{eff} varies between 1 and N where N is the number of particles. If N_{eff} is close to 1 which means the filter is facing severe degeneracy.

To avoid the degeneracy problem one can increase the number of particles but a better solution is to apply resampling. The resampling is used to eliminate the particles having small weights and focus on the particles with significant weights. Hence the particles with large weights will be selected and propagated to the next step.

Chapter 6

The Ensemble Kalman Particle Filter

6.1 Bridging the Ensemble Kalman and the Particle Filter

This chapter is based on [17, 18]. First we introduce a tuning parameter $\gamma \in [0, 1]$. This makes a continuous transition between the Ensemble and the Particle Filter update. We used Monte Carlo Simulations to find the optimum value of γ for the minimum root mean square error. Besides Particle filters, other Kalman filters as mentioned earlier work well under Gaussian noise. The updating scheme of the Ensemble Kalman filter consists of Ensemble Kalman filter updates based on a likelihood corrected by a Particle filter update. The advantage of the Ensemble Kalman Particle filter is that it is easy to implement these two steps and also the particle weights do not depending on the observation noise variables. By applying resampling, this filter avoids the particle ties. The Ensemble Kalman Particle filter does not require any structure for the state dynamics. The system can be stochastic or dynamic.

The tuning parameter $\gamma \in [0 , 1]$, which allows continous interpolation between the Ensemble Kalman filter and the Particle filter. The parameter γ controls the bias variance trade-off between a correct update and keeping the diversity of the sample.

First, consider the analysis scheme at a single fixed time of Ensemble Kalman Particle Filter (EnKPF) for the linear measurement function.

Assume the ensemble $\boldsymbol{X^f_i = (X^f_1, X^f_{2,} . . . . . . X^f_{i,} . . . X^f_N)}$ where i= 1,2,. . .,N with N the number of ensembles and Y, the observation data, is available.

6.2 Ensemble Kalman Particle Filter Algorithm

1) Compute the estimated prediction covariance [14]

   Covariance of the ensembles can be calculated using the ensemble mean

$$\overline{\boldsymbol{X^f}} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{X_i^f} \tag{6.1}$$

$$\overline{\boldsymbol{P^f}} = \frac{\boldsymbol{1}}{\boldsymbol{N-1}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{X_i^f} - \overline{\boldsymbol{X^f}})(\boldsymbol{X_i^f} - \overline{\boldsymbol{X^f}})^{\boldsymbol{T}} \tag{6.2}$$

2) Choose $\boldsymbol{\gamma \in [0 , 1]}$ apply the EnKF, based on the observation error covariance $\mathbf{R}/\boldsymbol{\gamma}$

Here $\boldsymbol{H}$ is used to represent the linear measurement function

$$K_1(\gamma)= P^f H^T (HP^f H^T + R/\gamma)^{-1} = \gamma P^f H^T (\gamma H P^f H^T + R)^{-1} \tag{6.3}$$

$$v_i = X_i^f + K_1(\gamma)(y_i - H(X_i^f)) \tag{6.4}$$

3) Compute Q as the process covariance

$$Q = \frac{1}{\gamma} K_1(\gamma) R K_1(\gamma)^T \tag{6.5}$$

4) Compute weights as

$$W_i = \Phi(y; Hv_i, \frac{R}{1-\gamma} + HQH^T) \tag{6.6}$$

Normalize the weights by diving by the weights with weight sum

$$\widehat{W_i} = W_i / \sum_{i=1}^{N} W_i \tag{6.7}$$

5) Choose indices by sampling from the weights with some balanced sampling.

The resampling index $\boldsymbol{S(i)}$ is chosen for each member v_i according to $\widehat{W_i}$ with residual resampling

6) To compute the updated state generate $\varepsilon_{1,j}$ in

$$X_i^u = V_{s(i)} + K_1(\gamma)\varepsilon_{1,j}/\sqrt{\gamma}\varepsilon_{1,j} \tag{6.8}$$

Here $\varepsilon_{1,j}$ is the random observation error drawn from the Gaussian N(0,R).

7) Compute $$K_2(1-\gamma) = (1-\gamma) QH^T ((1-\gamma) HQH^T + R)^{-1} \tag{6.9}$$

and generate $\varepsilon_{2,j}$ from N(0,R) and apply the EnKF with inflated observation error

$$X_i^a = X_i^u + K_2(1-\gamma)[y + \frac{\varepsilon_{2,j}}{\sqrt{1-\gamma}} - HX_i^u] \tag{6.10}$$

For the nonlinear measurement function is represented as h

$$K_1(\gamma) = \tag{6.11}$$

$$\{\frac{1}{N-1}\sum_{i=1}^{N}(X_i^f - \overline{X^f})[h(X_i^f) - h(\overline{X^f})]^T\} * \{\frac{1}{N-1}\sum_{i=1}^{N}(h(X_i^f) - h(\overline{X^f})[h(X_i^f) - h(\overline{X^f})]^T + R/\gamma)^{-1}\}$$

$$X_i^a = X_i^u + K_2(1-\gamma)[y + \frac{\varepsilon_{2,j}}{\sqrt{1-\gamma}} - h(X_i^u)] \tag{6.12}$$

Here matrix inversion is continuous therefore it is easy to check when γ tends to zero. Given $K_2(1-\gamma)$ is non zero, the particle filter update is obtained. When γ tends to 1 then $K_1(\gamma)$ exists so the Ensemble Kalman Filter is obtained [14].

Chapter 7

Noise

As discussed in the introduction the noises are uncertainties. There are many ways to model these uncertainties. In this thesis uncertainties are modeled probabilistically. Because the observation or measurements are impure and also contaminated by the unknown effects, these effects are known as noise. Even the best model may not perfectly predict the new measurements which are known as modeling errors bias. In Kalman filtering one of the important assumptions is that noise is modeled using a Gaussian distribution.

7.1 Gaussian Noise

Sensors are commonly used for signal measurement. Imperfect sensors have noise characteristics. This sensor noise is modeled as an additive Gaussian random variable. The Gaussian noise component having the pdf equal to the normal distribution, (also called Gaussian distribution). Let P of a Gaussian random variable of x be defined by [23]

$$\mathrm{P(x)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}} \qquad (7.1)$$

where μ is the mean value and **σ** is the standard deviation.

The exponential function $-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}$ is a quadratic function of the variable x, hence the parabola points downwards depending on the coefficient of the negative quadratic term. The term $\frac{1}{\sigma\sqrt{2\pi}}$ is independent of x, therefore assumed as a normalization factor [23].

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\left(-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}\right)\mathrm{dx} = 1 \qquad (7.2)$$

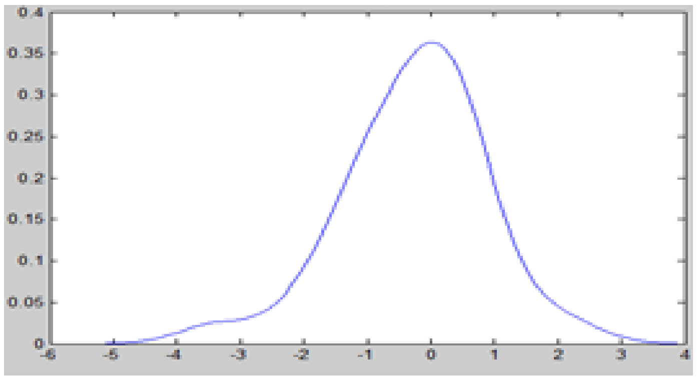

Figure 1: simulation result of generated zero mean Gaussian noise distribution μ=0, σ =1

7.2 Multivariate Gaussian distribution

7.2.1 Non zero mean Gaussian

A vector of random variable $x = [x_1, x_2, x_3 . . . x_n]^T$ is said to have Multivariate Gaussian distribution with mean μ, and covariance Σ. Its probability density function is defined by [23].

$$P(x) = \frac{1}{2\pi^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \qquad (7.3)$$

This can be represented as $x \sim N(\mu, \Sigma)$.

In the case of non-zero mean or multivariate Gaussian, the argument of exponential function

$-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$ is a quadratic form of the variable x. Since Σ is positive definite matrix, Σ^{-1} also a positive definite matrix [23]. For any vector $x \neq \mu$; $(x-\mu)^T \Sigma^{-1}(x-\mu)>0$ hence

$-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)<0$.

The coefficient term $\frac{1}{2\pi^{n/2}|\Sigma|^{1/2}}$ which is independent of x is then considered as normalization factor [23]

$$\frac{1}{2\pi^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} . . . \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) dx_1, dx_2, . . . dx_n = 1; \qquad (7.4)$$

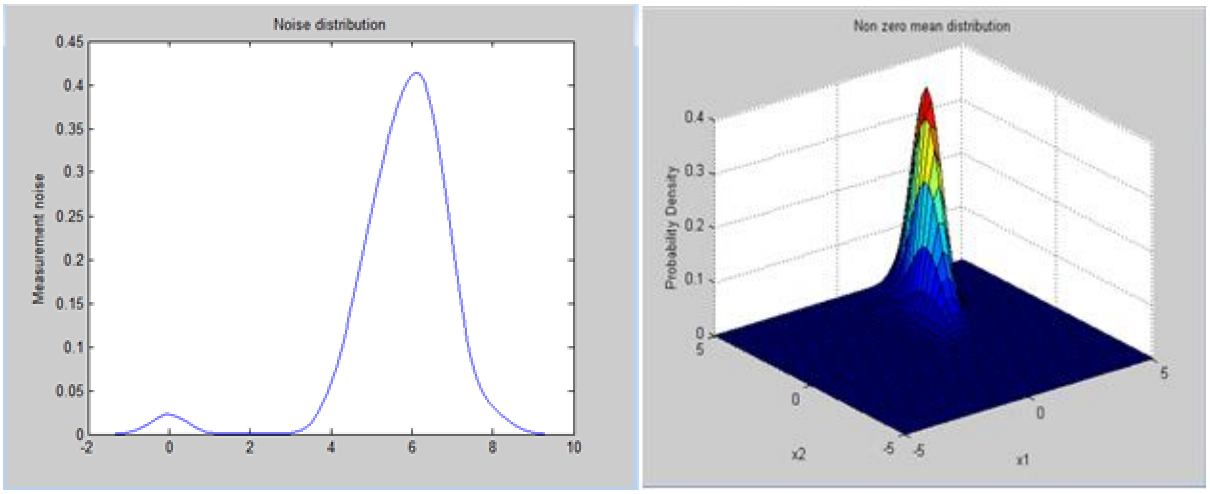

Figure 2: simulation result for the nonzero mean Gaussian

7.2.2 Gaussian with non-zero mean and Bias offset

As discussed earlier the inertial sensors like accelerometers and gyroscopes have errors due to the bias. Also, MEMS inertial sensors are affected by any change in physical properties like pressure, temperature or height leading to sensor bias. The bias can change the output value and providing additional error to the noise. The total bias of a sensor is defined as the average output signal which has no correlation with input signal related to acceleration or rotation. This means the bias offset can be defined as the value of output when the input is zero. This can be considered for an accelerometer by finding the acceleration when the object is not moving. Similarly for the Gyroscope the bias can be calculated by finding the angular rotation when the senor is not in rotation [24].

Simulation of Gaussian noise with mean = 5 and with bias of 10. The MATLAB code is given below.

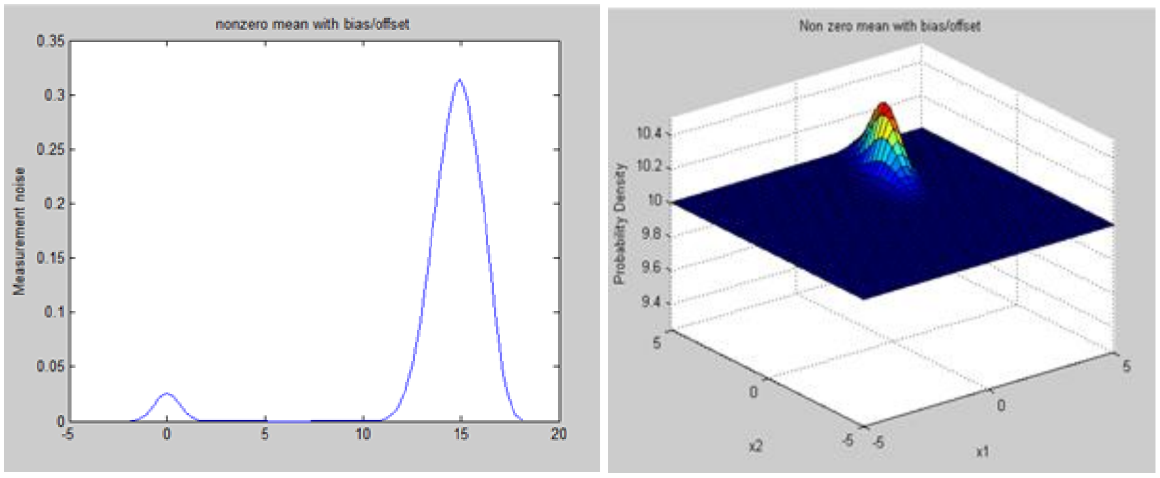

Figure 3: simulation result for Gaussian with non-zero mean and Bias offset

7.2.3 Gaussian with Non zero mean and drift

By considering the inertial sensors the input signals are time integrated therefore a constant bias of Ɛ causes an error which is growing linearly with time. For example an accelerometer x (t) = $\frac{1}{2}Ɛt^2$.

This means for 10 micro g bias in accelerometer the error in position results in 0.005m after10 seconds and 50m after 1000 seconds. This kind of accumulation of small bias over time is called drift [24].

Simulation of non- zero mean with drift

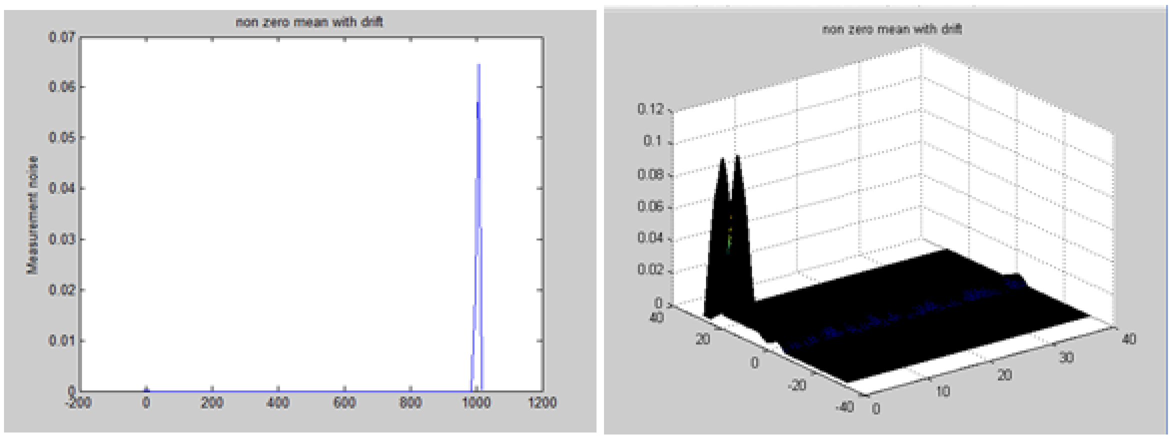

Figure 4: simulation result for Non zero mean with drift

7.3 Non Gaussian noise

Noise is an extra signal which interferes with the observation or output signal from the sensor. These noises are unknown and may be from another sensor or from the other sensor itself but noise is present in every sensor is often difficult to characterize. Therefore in this thesis work Gaussian noise is modified and transformed to non-Gaussian. MEMS sensors typically consist of angular random walk, rate random walk, velocity random walk, acceleration random walk etc. The position random walk is used to reduce the complexity in this work.

7.3.1 The transformation of Gaussian to non- Gaussian

With mean = 10 variance = 4 and added skewness = 0.5 (for a Gaussian noise it will be 0). The skewness is a measure of symmetry. If a distribution is symmetric it looks the same to the both sides of the center point. Finally the kurtosis = 4 (which is equal to 3 for purely Gaussian). Kurtosis is similar to skewness. If is a measure of “tailedness” of the probability distribution [27].

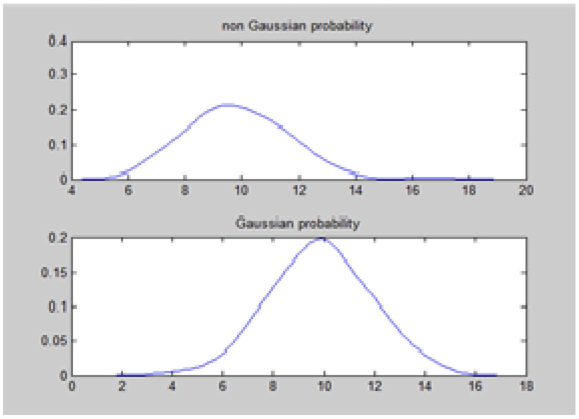

Figure 5: the transformation of Gaussian to non-Gaussian distribution

7.3.2 The simple Random walk

Suppose that X= (X_1, X_2, X_3…..) is a sequence of independent random variables, and each variables having values 1 and -1 with probabilities p∈ [0,1] and 1-p respectively. Let Y= (Y_0, Y_1, Y_2, ..) be partial sum processes associated with X so that

$$Y_n = \sum_{i=1}^{n} X_i, \qquad (7.5)$$

Where n ∈ N and N is the number of particles [25].

The sequence Y is the simple random walk with parameter p. Consider a particle on an axis at each discrete time step the particle either one unit to the left (with probability p) or to the right(with probability 1-p), independently step to step. In this thesis work it is assumed that each step occurs with equal probability, i.e. p = 0.5.

The expected value and standard deviation of sequence X are

$$E(X) = 2p-1 \qquad (7.6)$$

$$Var(X) = 4p(1-p) \qquad (7.7)$$

Y_n has probability density function [25]

$$P(Y_n=k) = \binom{n}{(n+k)/2} p^{(n+k)/2} (1-p)^{(n-k)/2}, \ k \in \{-n,-n+2, \ldots n-2, n\}, \qquad (7.8)$$

The mean and variance of Y_n are [25]

$$E(Y_n) = n(2p-1)$$

$$Var(Y_n) = 4np(1-p),$$

In this work simulation of random walk considered length of step space 0.1, length of time step 0.01, number of time step 40, number of particles 100, and probability to move left and to move right as 0.5.

7.3.3 Simulation of simple Random Walk

Noise distribution of the random walk

The Probability distribution of a Random walk is generated with Gaussian distribution and with added mean = 10 variance 4 and added skewnes 0.5 (for a purely Gaussian noise it will be 0) and finally with the kurtosis 4 (which is equal to 3 for Gaussian) to get non-Gaussian noise.

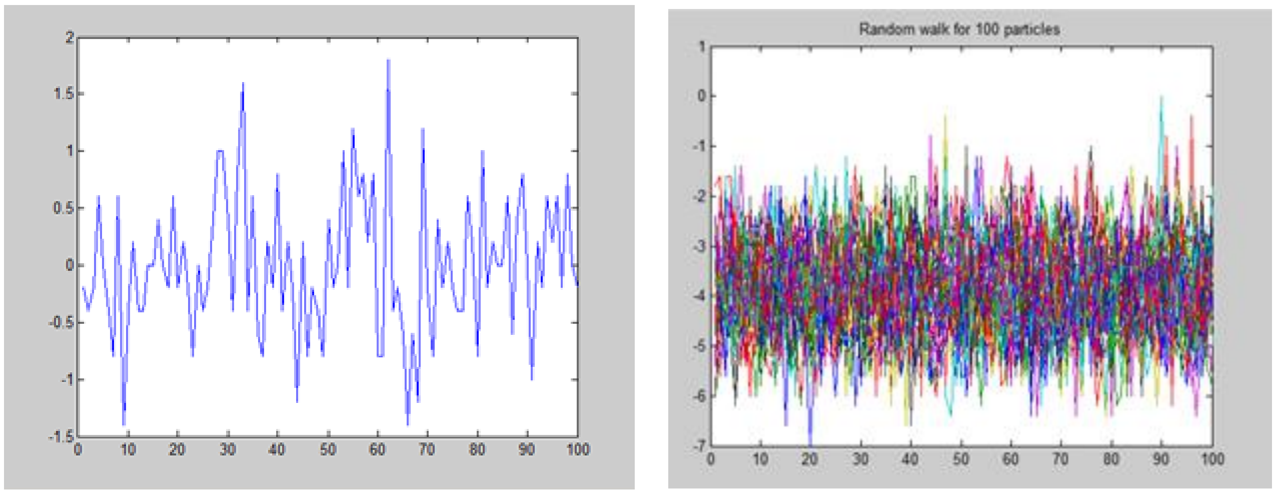

Figure 6: simulation result for the simple random walk

7.3.4 Simulation of the probability density of transformed Gaussian Random Walk

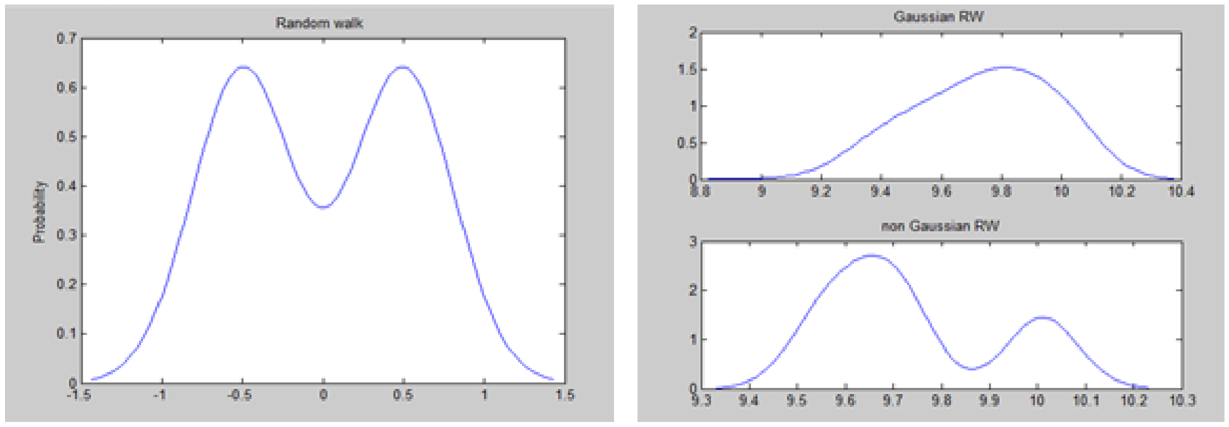

Figure 7: Simulation of the probability density of transformed Gaussian random walk

7.4 Laplacian noise

Due to the unknown noise in the environment the simplest approach to add noise is Laplacian noise. This is also called noise with bi-exponential distribution which means that two exponential functions are utilized back to back, one is positive and one is negative. Therefore it has two parameters. One is μ (the position) and b which is the scale (the spread). [26]

$$f(x|\ \mu,b) = \frac{1}{2b}\exp\ \frac{-(|x-\mu|)}{b} \qquad (7.9)$$

Generating of Laplacian noise requires transforming the Gaussian noise using a nonlinear memory-less transformation. Use the transformation of x as x= $F^{-1}(w)$ where F is the cumulative distribution of the Laplacian pdf and w is the uniform random variable of the interval [0, 1]. The MATLAB code for generating Laplacian noise is given in the appendix.

7.4.1 Simulation results of the Laplacian noise

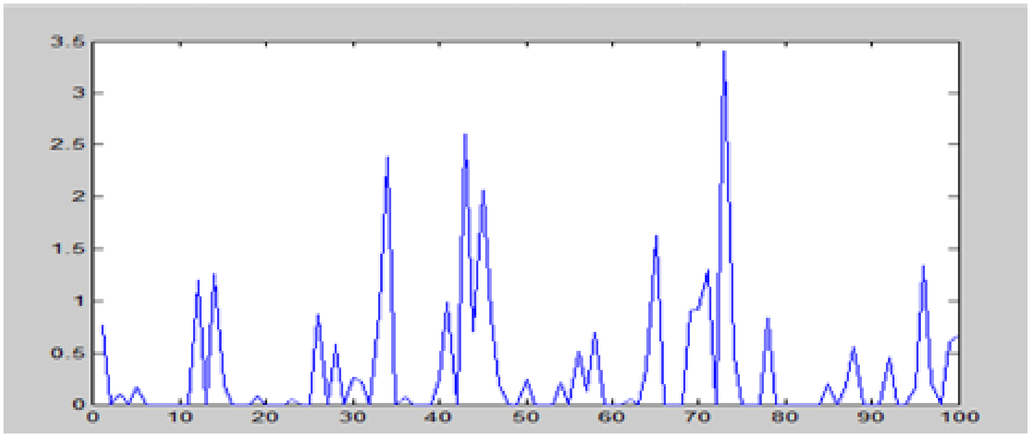

Figure 8: Laplacian noise

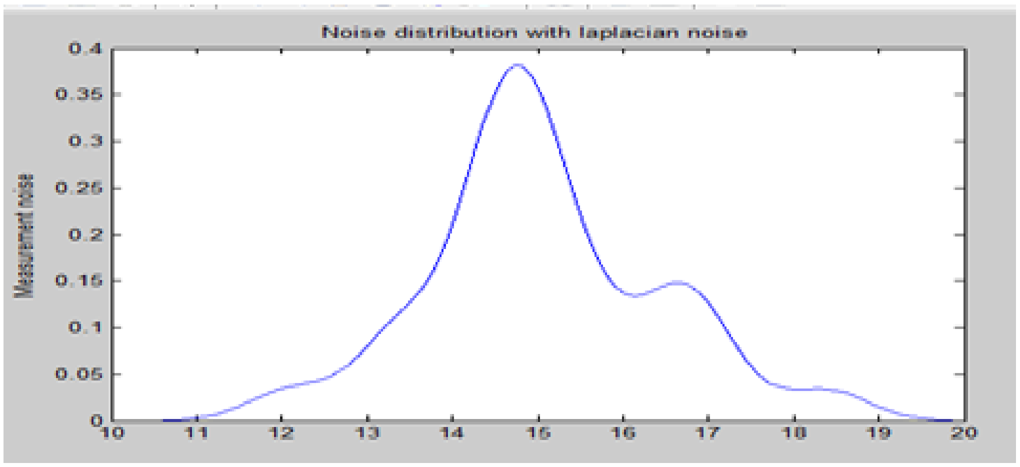

Figure 9.1: Laplacian noise distribution

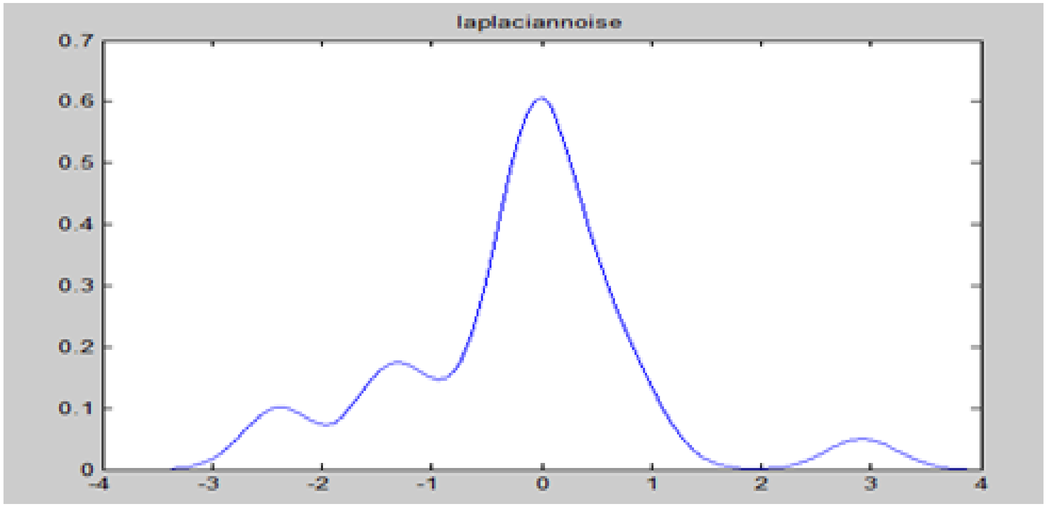

Figure 9.2: Laplacian noise distribution

Chapter 8

Simulation Results

This chapter describes a few simulation results where the states of the aforementioned system were estimated by the Particle filter, Ensemble Kalman filter and the Ensemble Kalman Particle fiter. Consider a system where the state variable $X_k$is one dimensional andits nonlinearly connects the observation vector $Y_k$whichis also one dimensional. The state variable of the system under the noise is given by

$$X_k = \frac{1}{2} X_{k-1} + \sin(1.2*k) + U_k \quad (8.1)$$

$$Y_k = X_k^2 + V_k \quad (8.2)$$

The state is initialized to $X_0 = [1,2]$

In order to compare the Ensemble Kalman Particle Filter with Ensemble Kalman Filter and Particle filter the norm of RMSE was computed.

The number of ensembles and number of steps are set as 100 and 40 respectively for all the filters in this thesis.

8.1 Norm. RMSE for different filters

Filters	$[\tau_1 , \tau_2]$	Norm. rmse
EnKPF	$[0.2 , 0.4]$ $\gamma=0.35$ $[0.4 , 0.6]$ $\gamma=0.5$ $[0.6 , 0.8]$ $\gamma=0.5$	1.8532 0.8819 0.9009
EnKF		3.276
PF		1.4815

Table1: Norm.rmse for EnKF, PF and EnKPF under non zero mean Gaussian

$\tau \in [\tau_1 , \tau_2]$ is a key factor to select the gamma value, which is the tuning parameter to bridge the Ensemble Kalman Filter and Particle Filter as mentioned in chapter 6. The constrained diversity interval $[\tau_1 , \tau_2]$ for different values the norm of rmse is less than EnKF and PF for all the simulations. From the table 1 the optimal performance of the EnKPF γ is 0.5the optimal gamma value under quasi-Gaussian with drift

Filters	[τ_1 , τ_2]	Norm. rmse
EnKPF	[0.2 , 0.4] γ=0.35 [0.4 , 0.6] γ=0.5 [0.6 , 0.8] γ=0.55	0.8909 0.90844 0.9104
EnKF		2.262
PF		1.7393

Table 2 :Norm.rmse for EnKF, PF and EnKPF under quasi-Gaussian with drift

From the above table when γ is close to zero this filter will work as Particle filter and γ is close to one it works as Ensemble Kalman Filter. From the tables the optimum value for γ is 0.5 for quasi-Gaussian noise.

8.2 Monte Carlo Simulation results

8.2.1 The optimal value gamma under Gaussian noise

Number of samples	Gamma value
10	0.6538
100	0.87003
500	0.88017
1000	0.51082

Table 3 : The value of gamma under Gaussian noise

8.2.2 The optimal value of gamma under Quasi-Gaussian noise

Number of samples	Gamma value
10	0.60669
100	0.5409
500	0.52248
1000	0.36328

Table 4: The value of Gamma under Quasi-Gaussian noise

8.2.3 The optimal value of gamma under drift

Number of samples	Gamma value
10	0.20659
100	0.21999
500	0.32898
1000	0.298857

Table 5: The value of gamma under drift

8.2.4 The optimal value of gamma under quasi-Gaussian Noise with random walk

Number of samples	Gamma value
10	0.28504
100	0.25032
500	0.27327
1000	0.31694

Table 6: The value of gamma under random walk

From these simulation result it is evident that under Gaussian noise EnKPF works as an Ensemble Kalman filter and under the quasi-Gaussian with drift it works similar to the Particle filter. The tuning parameter can take any value in the interval [0,1]. But from these table for quasi-Gaussian with drift or random walk γ value is close to 0 therefore it shows resembles a Particle Filter.

8.3 Simulation under different noise conditions

The simulation results of the Ensemble Kalman Filter, the Particle filter and the Ensemble Kalman Particle filter under Gaussian and quasi-Gaussian noise conditions are given below. The generated noise is added to the measurements of the model and simulated on MATLAB for 100 number of ensembles or particles and 40 iterations.

8.3.1 Under zero mean Gaussian noise

The zero mean noise is discussed in the chapter 7 in section 7.1. The Matlab source code is provided in the appendix.

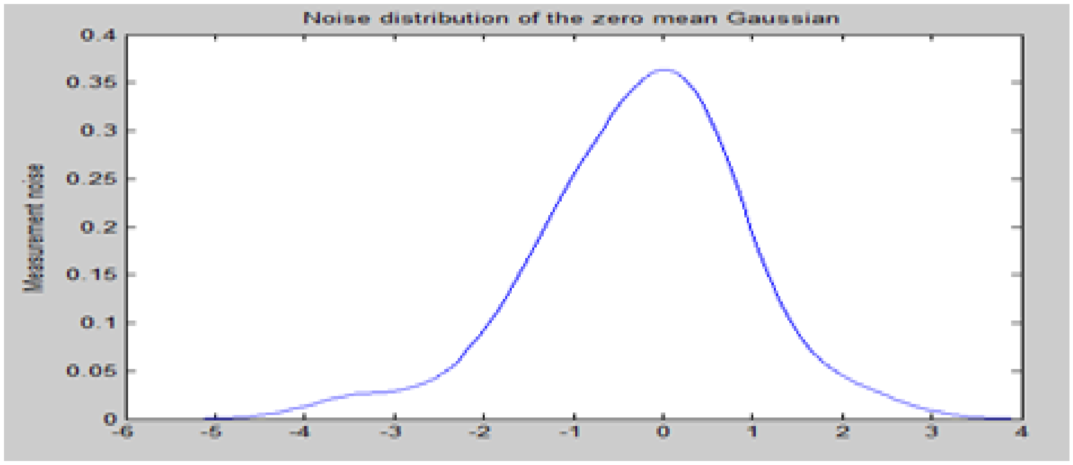

Figure 10. The Gaussian distribution of the measurement noise

The estimated state is computed using updated equations of the algorithm which is aforementioned in the previous chapters. The Matlab source code is included in the appendix.

8.3.2 State and estimated state of the system for EnKF,PF,EnKPF under Gaussian noise

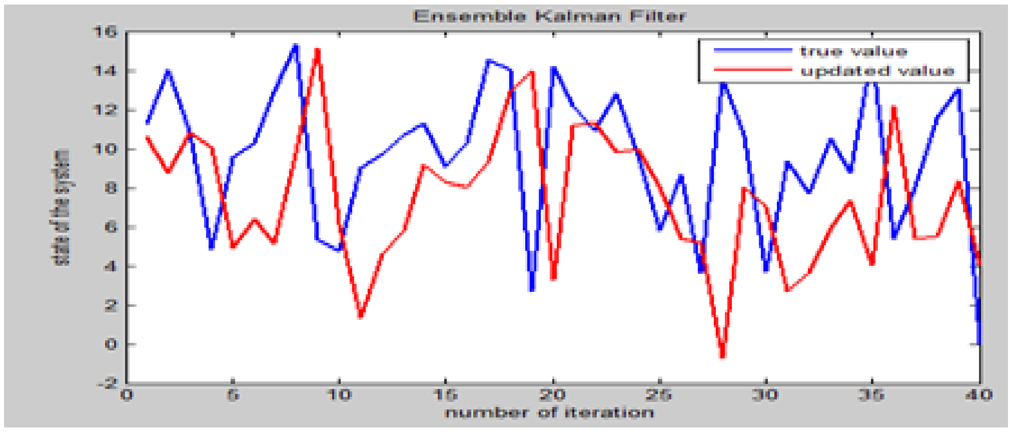

Figure 11.1: State and estimated state of EnKF under Gaussian

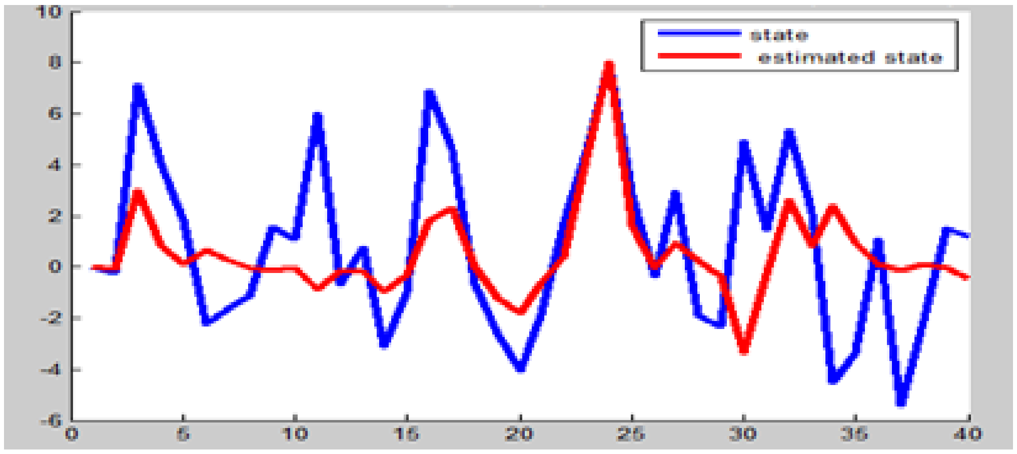

Figure 11.2: state and estimated state of PF under Gaussian

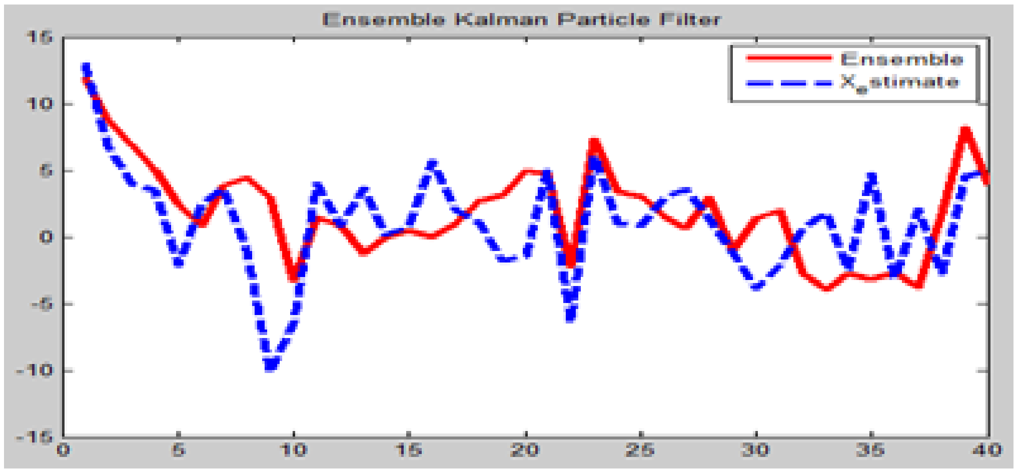

Figure11.3 : state and estimated state of EnKPF under Gaussian

8.3.3 Observation and filtered observation for EnKF,PF,EnKPF under Gaussian noise

The below shown simulation results of observation of the model and the filtered observation, which is computed using the estimated state.

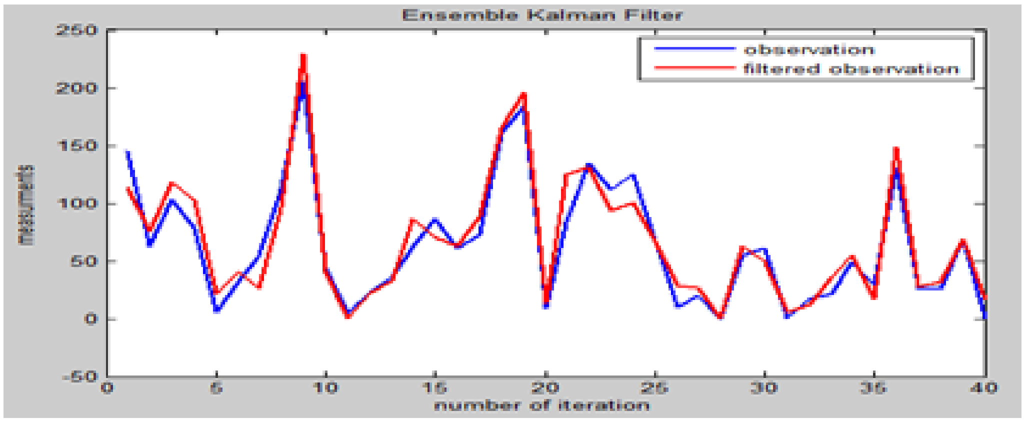

Figure 12.1: observations for EnKF under Gaussian

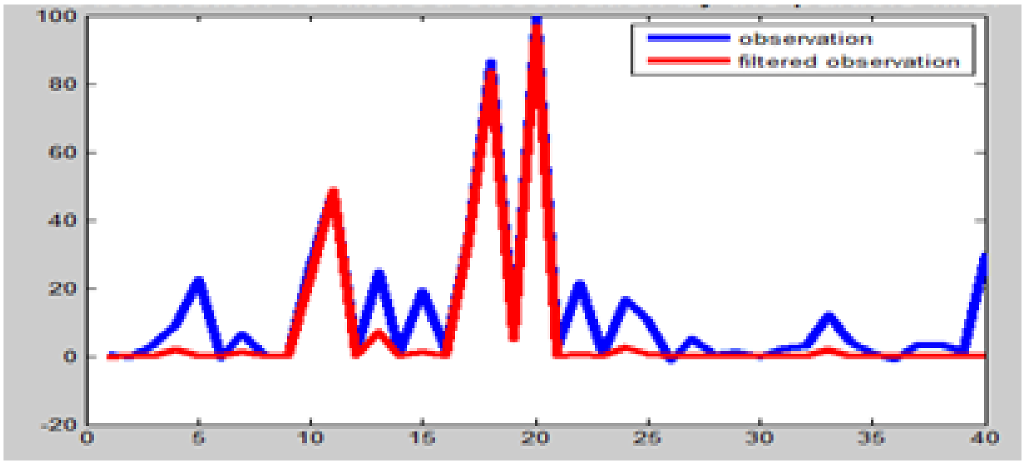

Figure 12.2 : observations for the PF under Gaussian

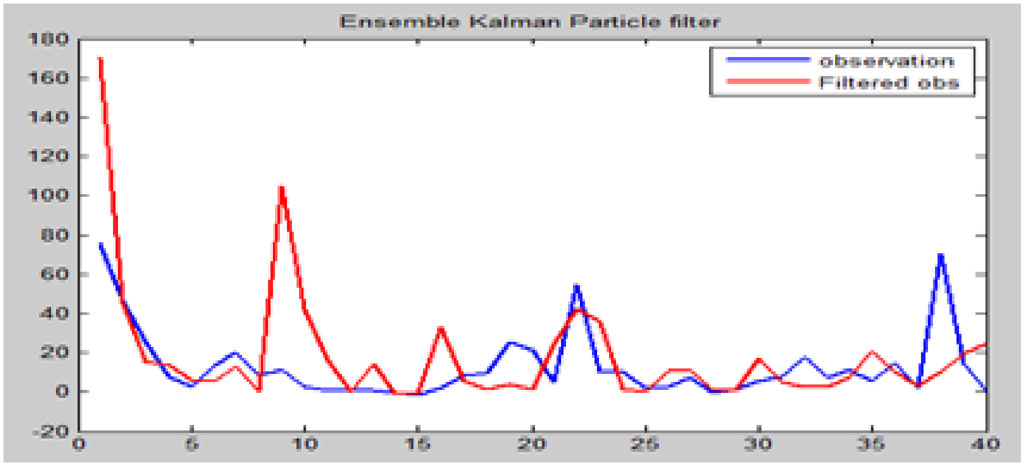

Figure 12.3 : the observation for the EnKPF under Gaussian

8.3.4 Root mean square error for the Filters under Gaussian

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	0.43577	4.7021	0.93193

Table 7 :Rmse for the filters under Gaussian condition.

From the table 7 it is evident that the Ensemble Kalman filter provides optimal solution under Gaussian noise.

8.4 Multivariate Gaussian Distribution

In this simulation the measurement noise generated with mean = 5 and the variance is 10

8.4.1.The multivarate Gaussian Distribution of the measurement noise

The multivariate Gaussian is discussed in the chapter 7 in section 7.2.1. The Matlab source code is provided in the appendix.

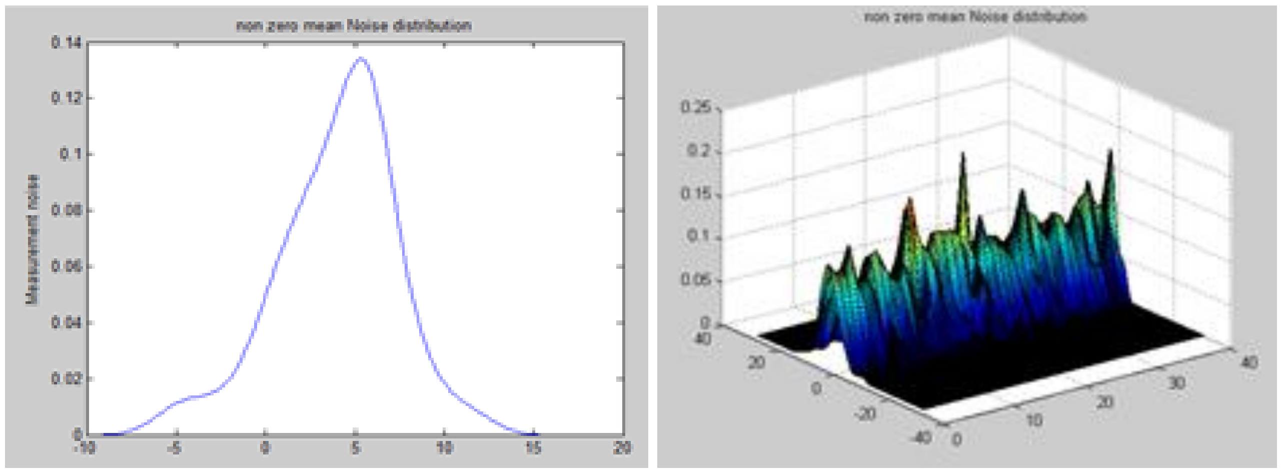

Figure 13: the multivariate Gaussian Distribution

8.4.2. State and estimated state for EnKF, PF, and EnKPF under Multivariate Gaussian

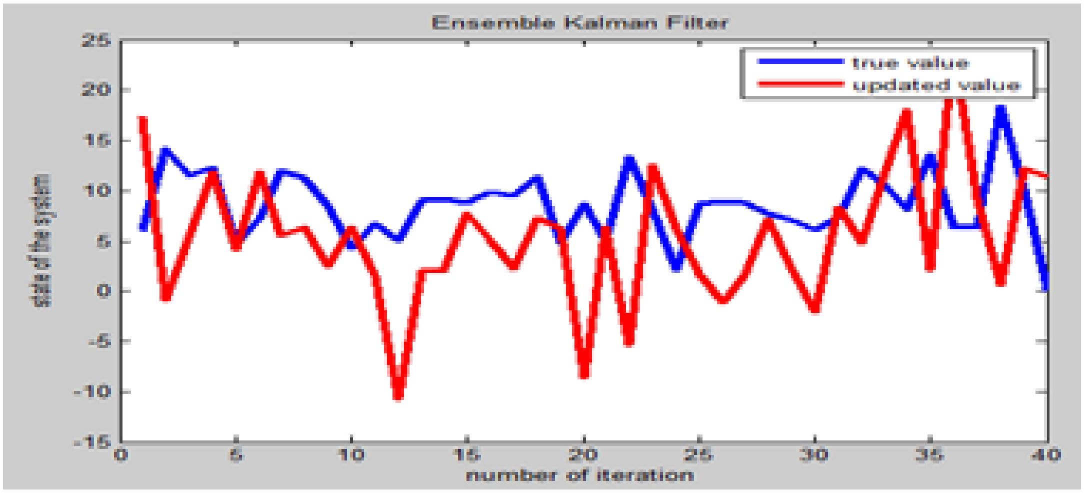

Figure 14.1: State and estimated state of EnKF under multivariate Gaussian

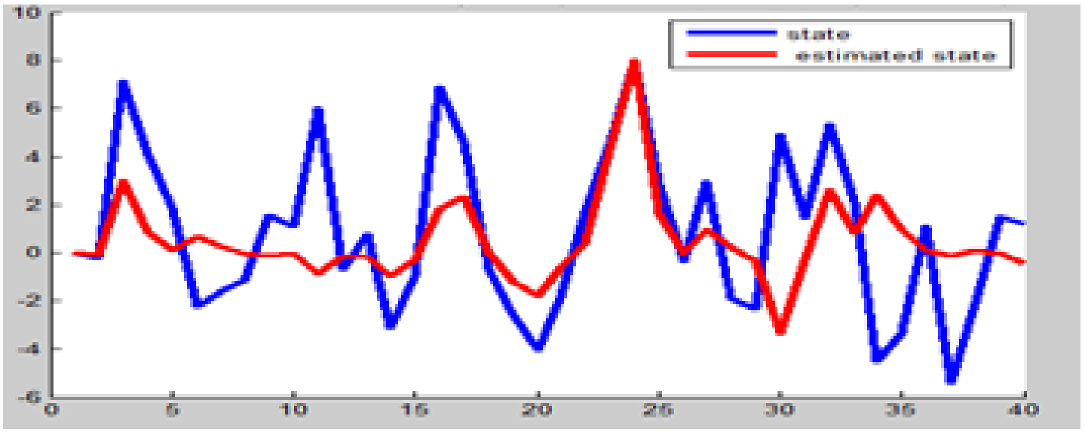

Figure 14.2: state and estimated state of PF under multivariate Gaussian

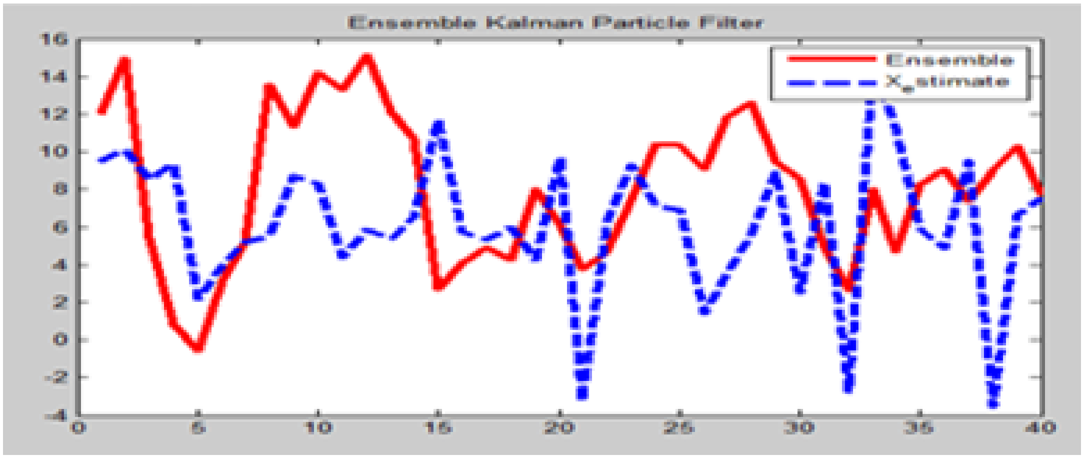

Figure 14.3: state and estimated state of EnKPF under multivariate Gaussian

8.4.3. The obseravation and filtered observation for EnKF, PF and EnKPF

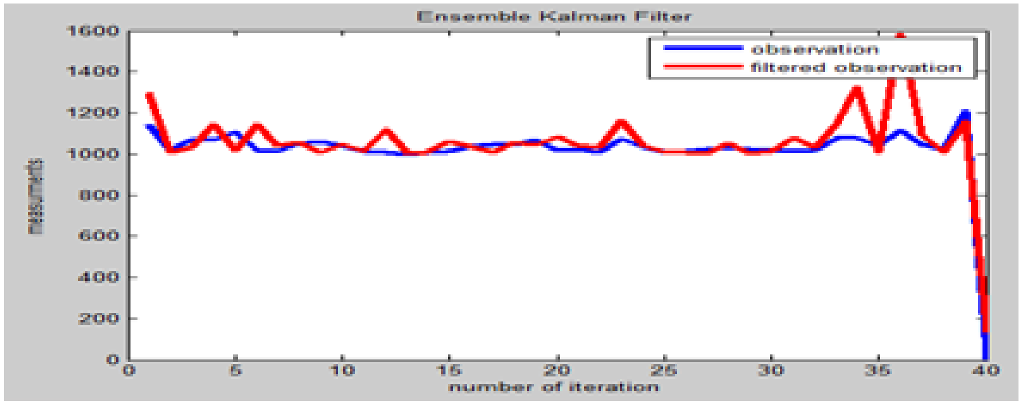

Figure 15.1: observations for EnKF under multivariate Gaussian

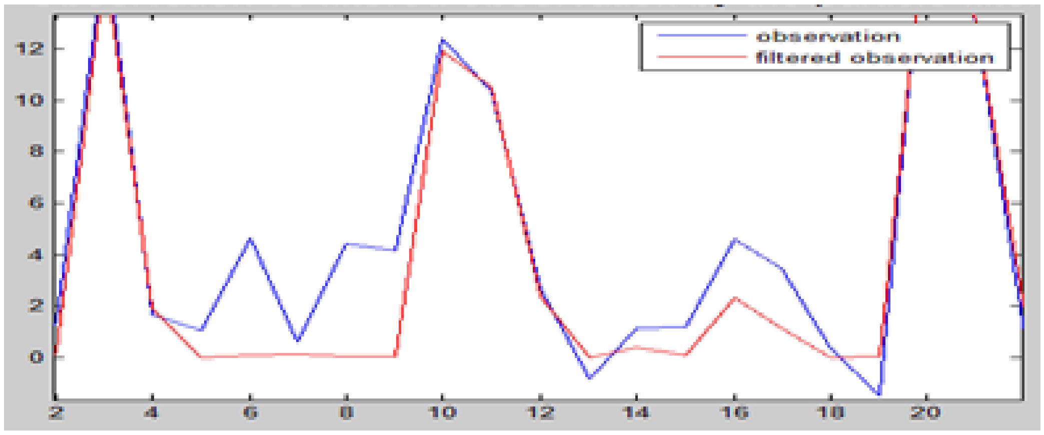

Figure 15.2: observations for the PF under multivariate Gaussian

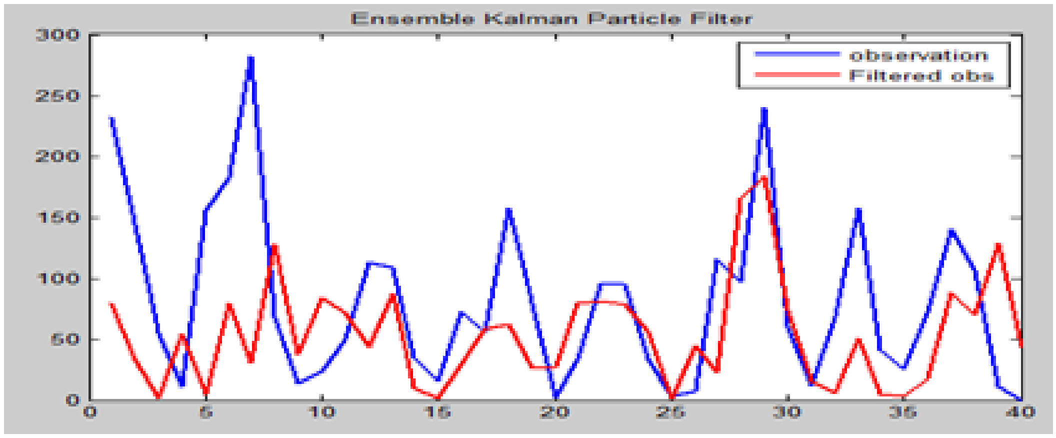

Figure 15.3: the observations for the EnKPF under multivariate Gaussian

8.4.4. Root mean square error for the Filters under Multivariate Gaussian

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	2.8164	1.4815	0.8782

Table 8 :Rmse for the filters under Multivariate Gaussian

The Ensemble Kalman Particle Filter is comparatively gives the optimum solution because of the less norm rmse.

8.5 Simulations of filters under Gaussian with non-zero mean and Bias offset

The measurement noise is generated as Gaussian with mean 5 and variance 10 and added a bias offset 10

8.5.1.The Gaussian with non-zero mean and Bias offset distribution

The Gaussian with non-zero mean and bias offset is discussed in the chapter 7 in section 7.2.2. The Matlab source code is provided in the appendix.

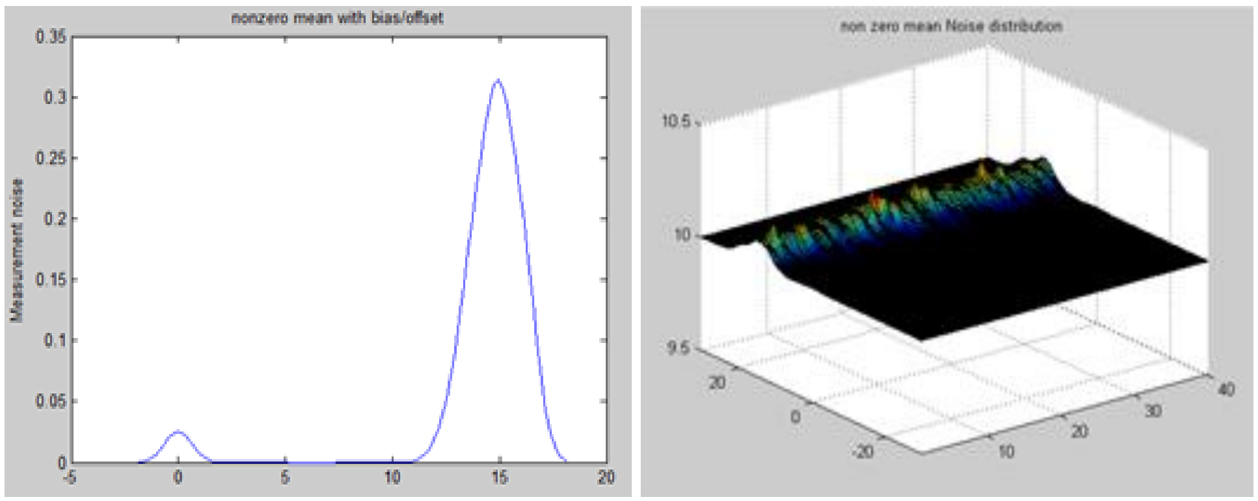

Figure 16: The Gaussian with non-zero mean and Bias offset distribution

8.5.2. State and estimated state for EnKF, PF, and EnKPF under quasi-Gaussian with bias

The state and estimated state under quasi-Gaussian noise with bias offset

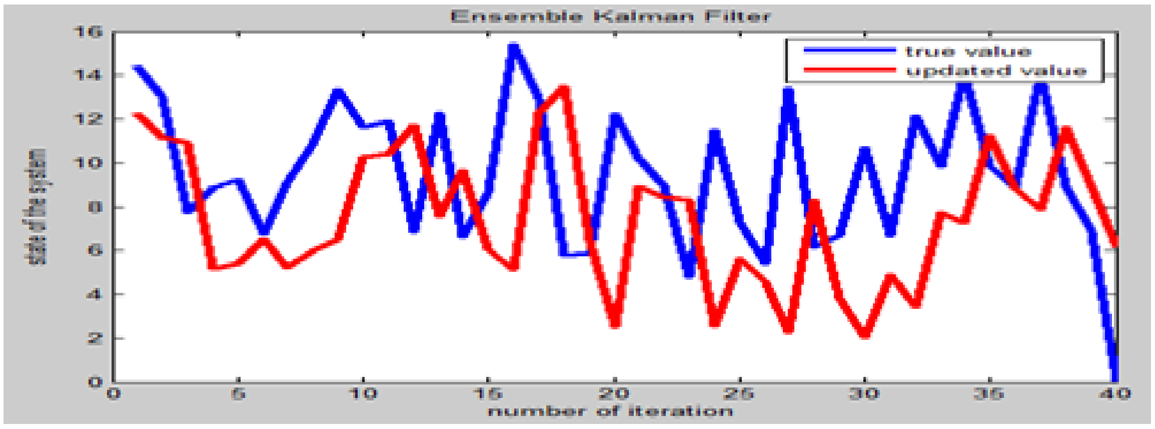

Figure 17.1: State and estimated state for EnKF under quasi-Gaussian noise with bias offset

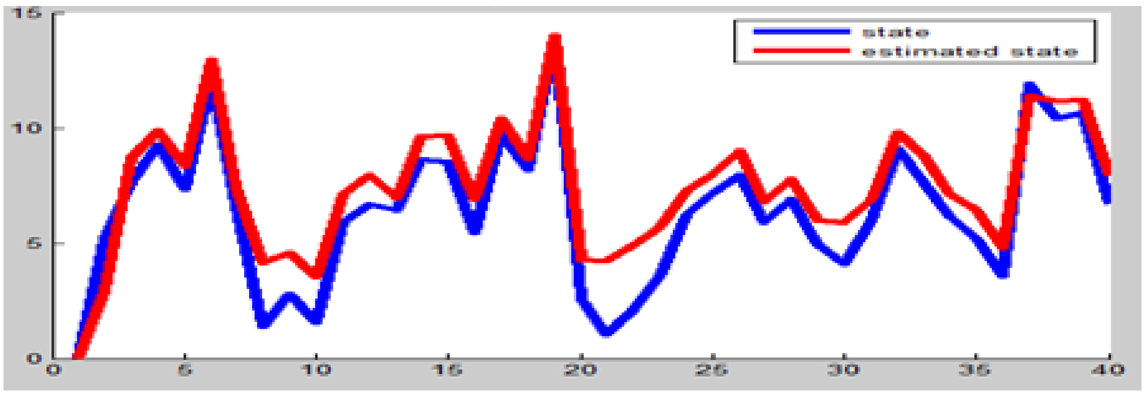

Figure 17.2: State and estimated state for PF under quasi-Gaussian noise with bias offset

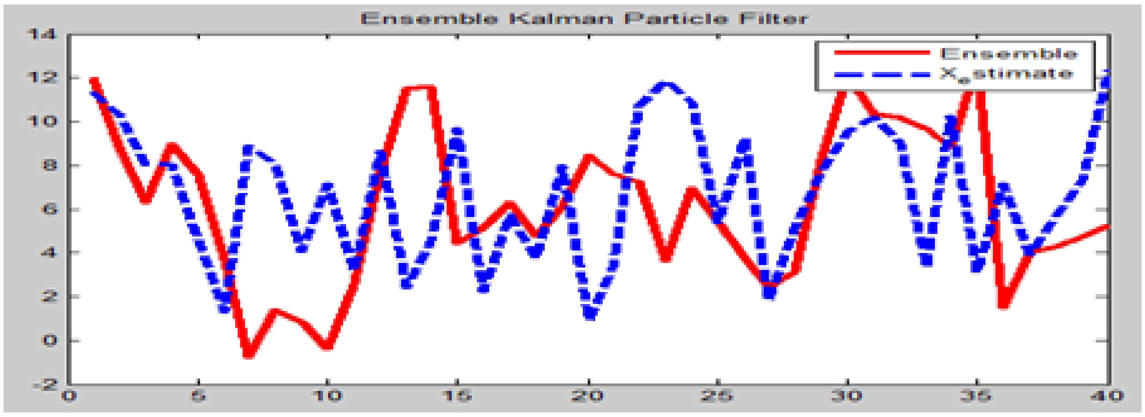

Figure 17.3: State and estimated state for EnKPF under quasi-Gaussian noise with bias offset

8.5.2. The observation and the filtered observation

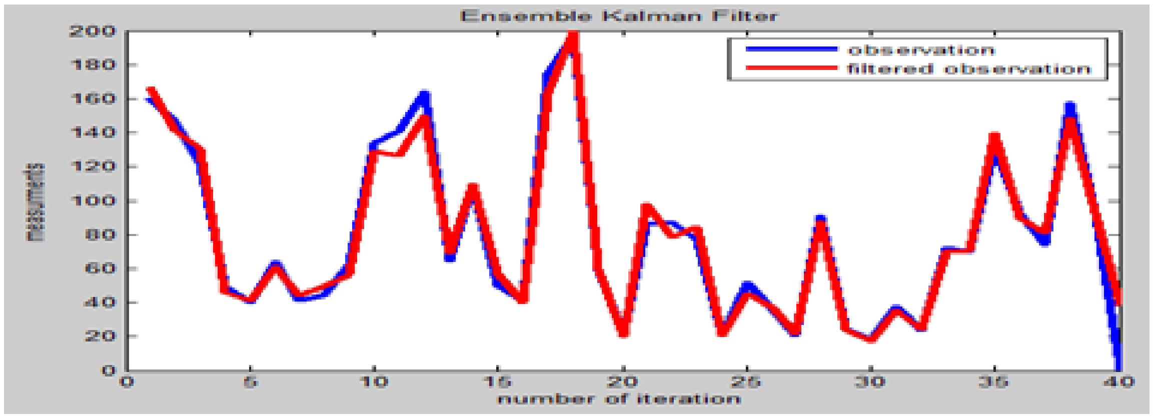

Figure 18.1: Observation for EnKF Figure under quasi-Gaussian noise with bias offset

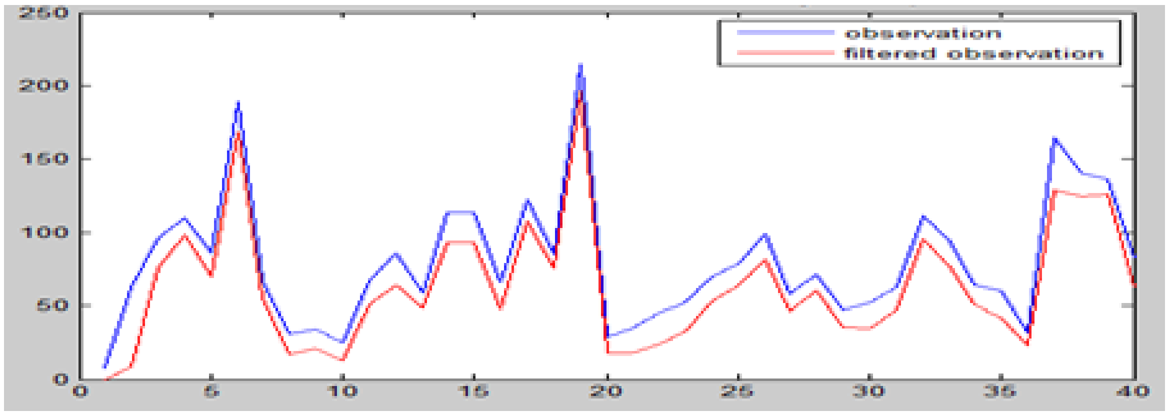

Figure 18.2: Observation for PF under quasi-Gaussian noise with bias offset

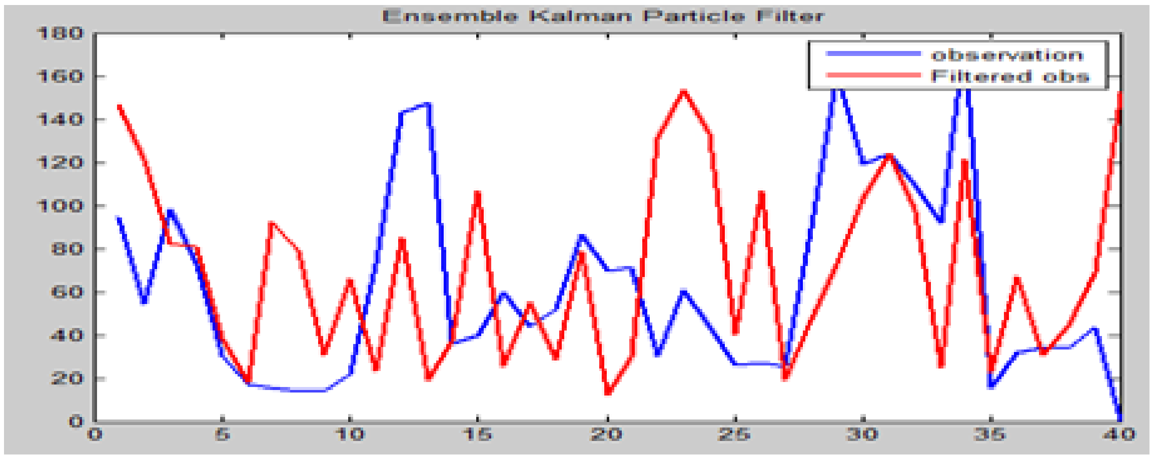

Figure 18.3: Observation for EnKPF under quasi-Gaussian noise with bias offset

8.5.3 Root mean square error for the Filters under quasi-Gassian with bias

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	2.5662	1.7393	0.87211

Table 9 : Filters under quasi-Gassian with bias

8.6 Simulations of filters under quasi-Gaussian with drift

The measurement noise is generated as Gaussian with mean = 5 and variance = 10 with drift =10t

The quasi-Gaussian noise with drift is discussed in the chapter 7 in section 7.2.3. The Matlab source code is provided in the appendix.

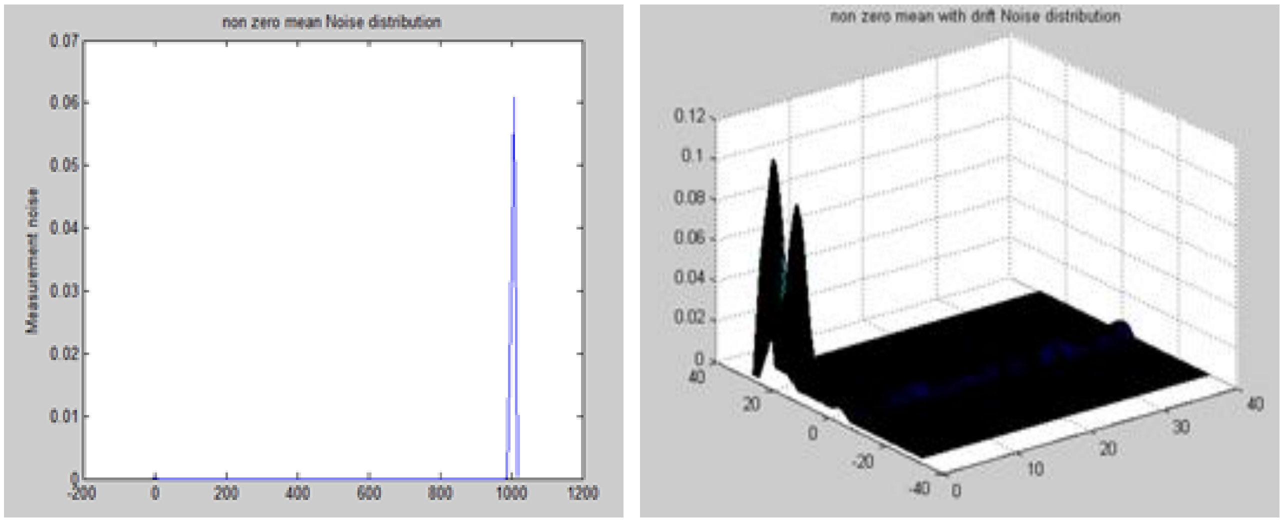

Figure 19: The Quasi-Gaussian with drift

8.6.1 The state and estimated state under Quasi-Gaussian with drift

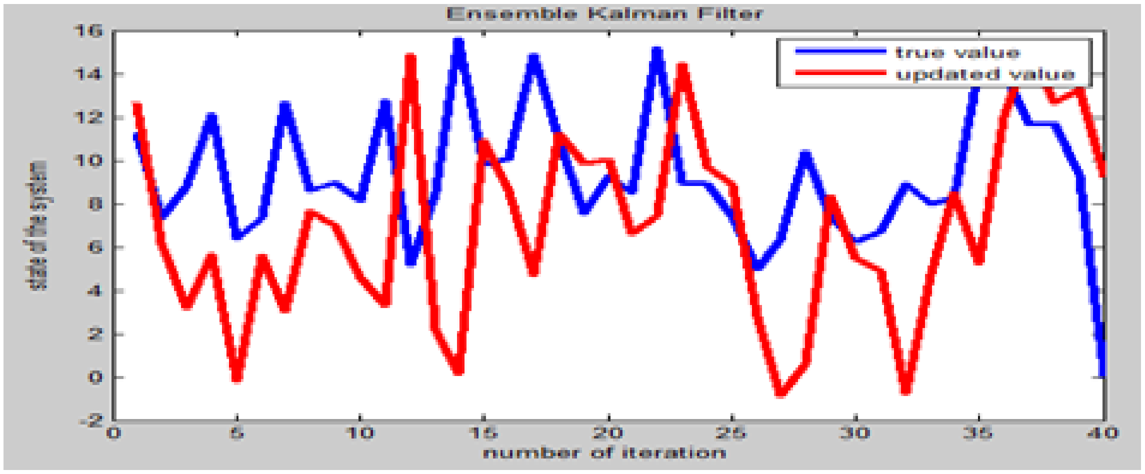

Figure 20.1 : State and estimated state for EnKF under Quasi-Gaussian with drift

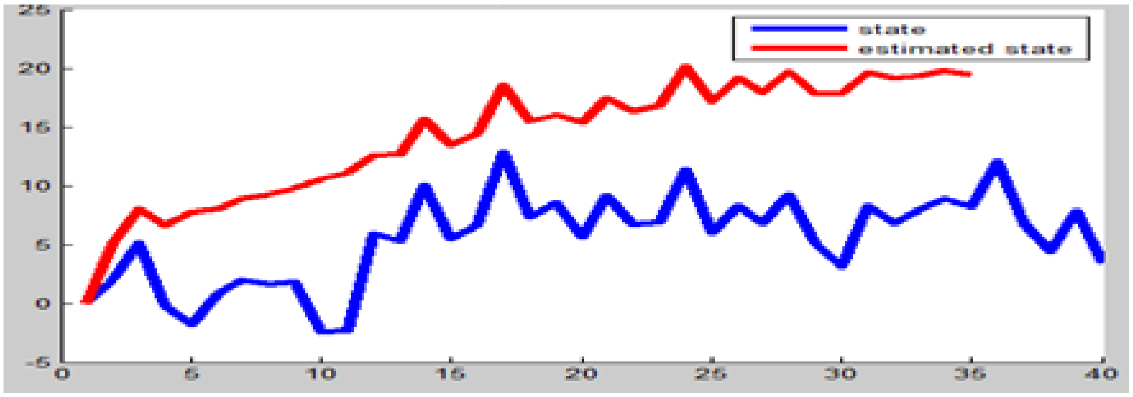

Figure 20.2 : State and estimated state for PF under Quasi-Gaussian with drift

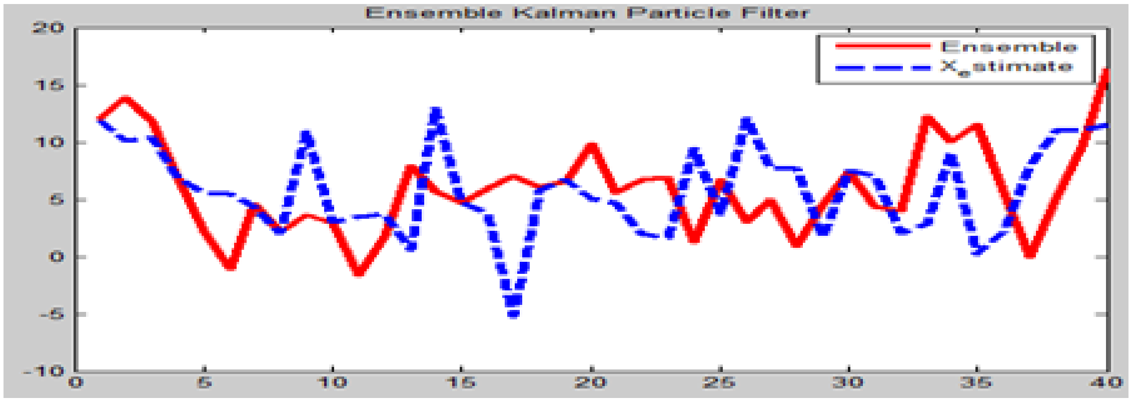

Figure 20.3 : State and estimated state for EnKPF under Quasi-Gaussian with drift

8.6 .2 The observation and filtered observation under Quasi-Gaussian with drift

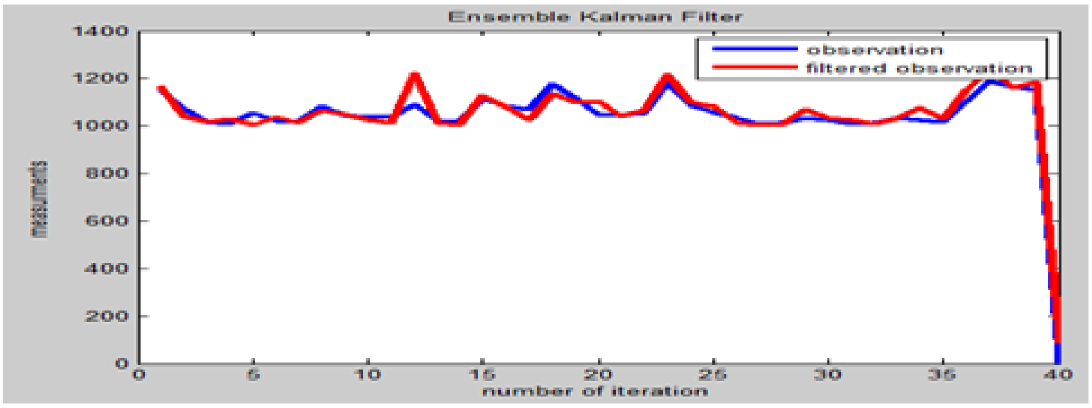

Figure 21.1: Observations for EnKF under Quasi-Gaussian with drift

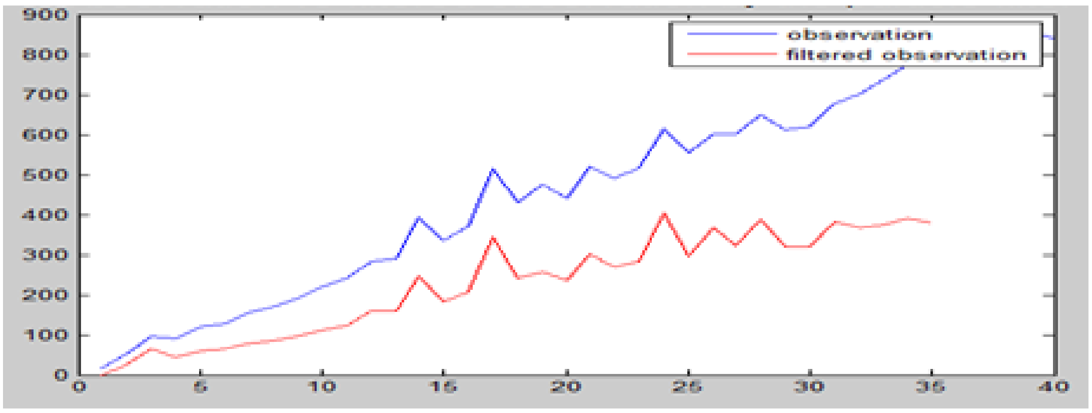

Figure 21.2 : Observations for PF under Quasi-Gaussian with drift

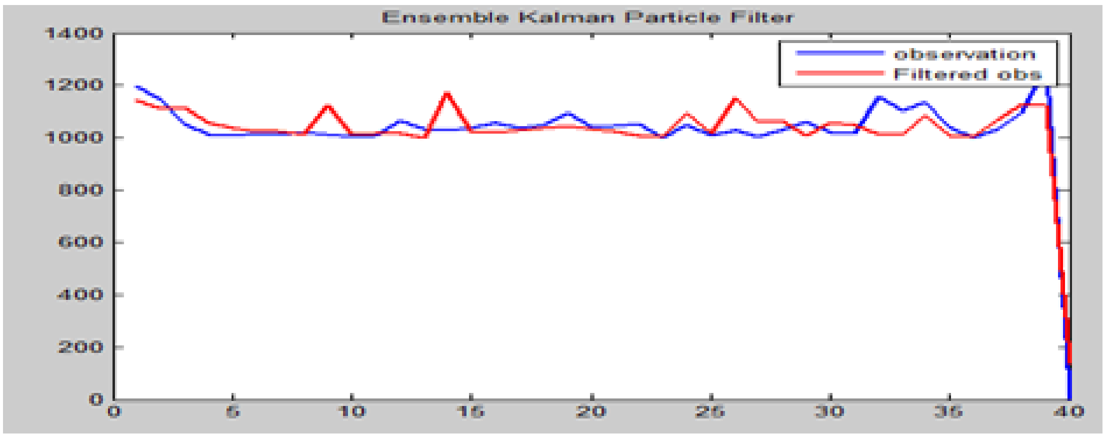

Figure 21.3 : Observations for EnKPF under Quasi-Gaussian with drift

8.6 .3 Root mean square error for the Filters under quasi-Gaussian with drift

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	3.1916	2.4096	0.91985

Table 10 : Filters under quasi-Gassian with drift

8.7 Random walk

The Random Walk is discussed in the chapter 7 in section 7.3.2. The Matlab source code is provided in the appendix.

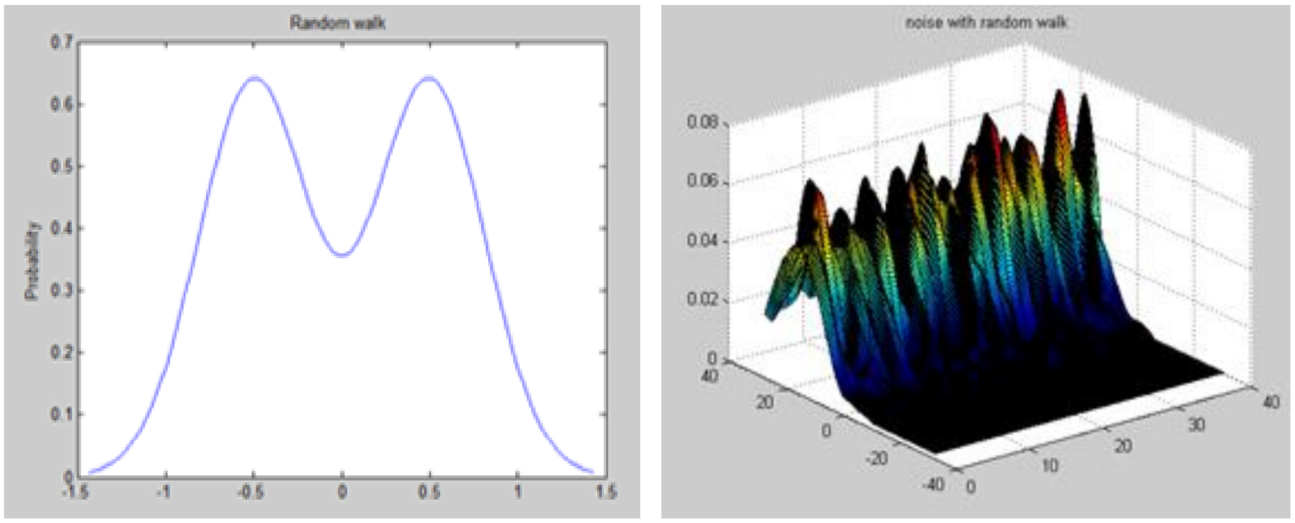

Figure 22 : Random walk distribution

8.7.1. State and estimated states under random walk

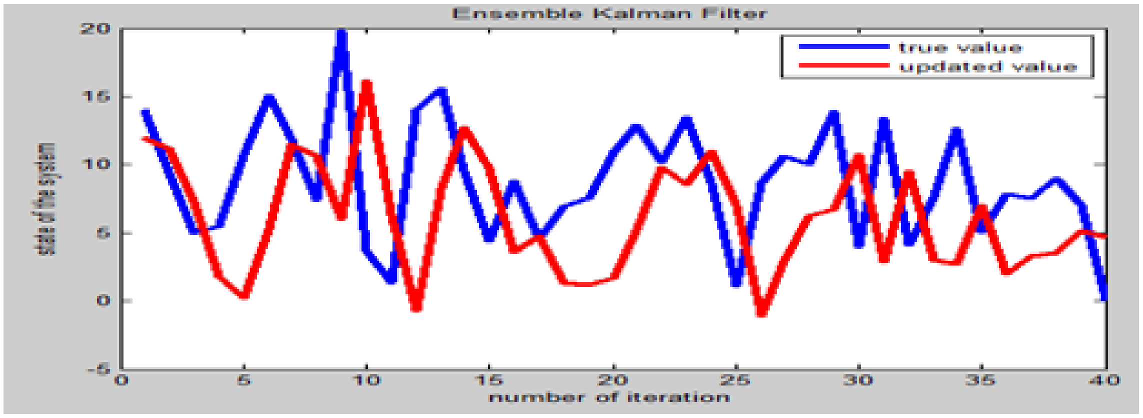

Figure 23.1: State and estimated state for EnKF under random walk

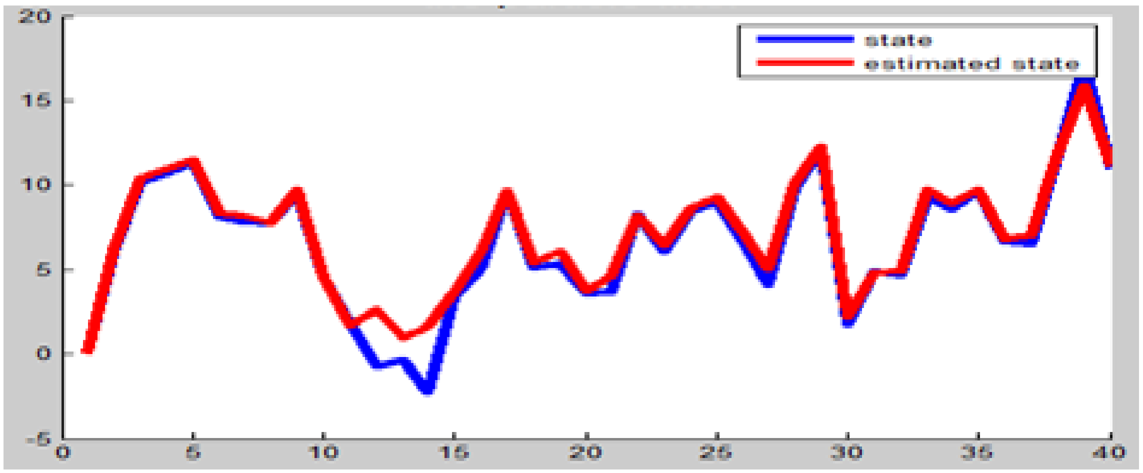

Figure 23.2 : State and estimated state for PF under random walk

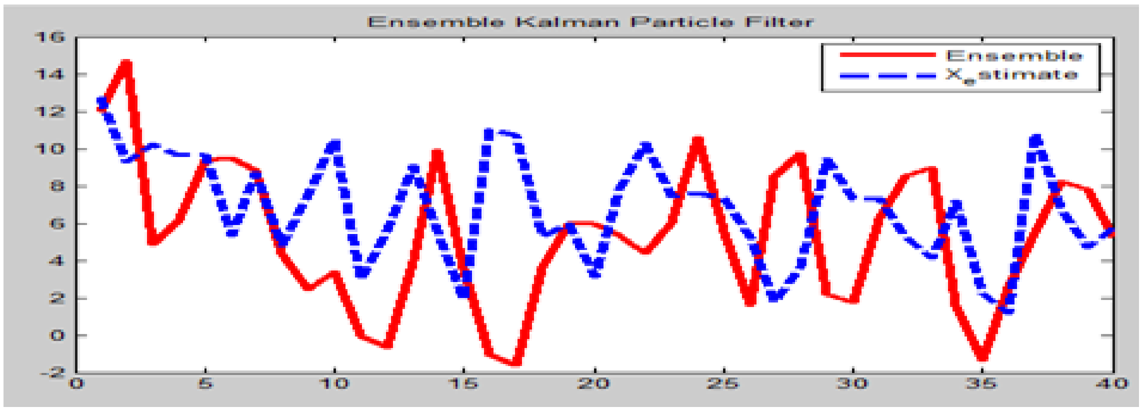

Figure 23.3 : State and estimated state for EnKPF under random walk

8.7.2 Observation and filtered observation under random walk

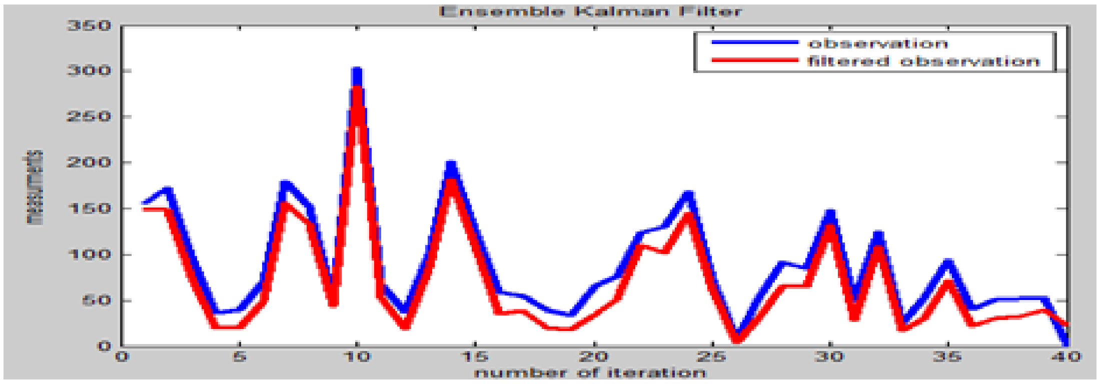

Figure 23.1: Observations for EnKF under random walk

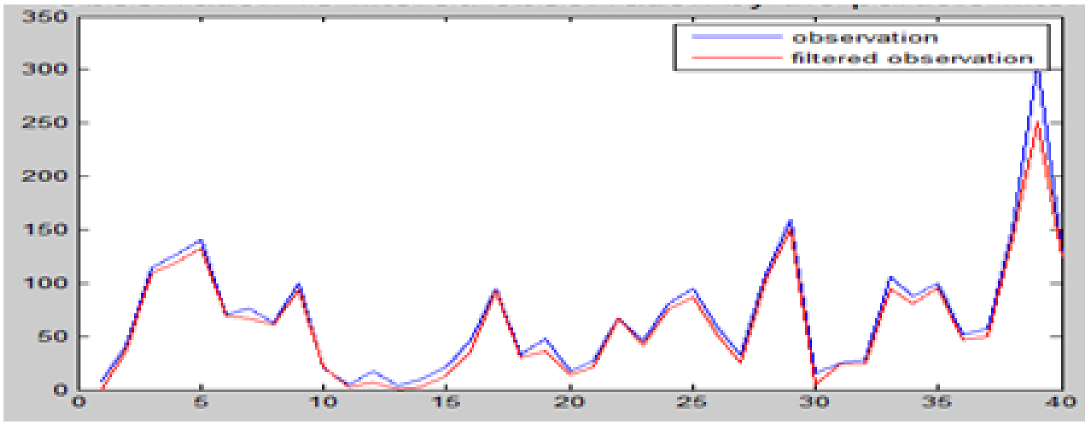

Figure 23.2: Observations for PF under random walk

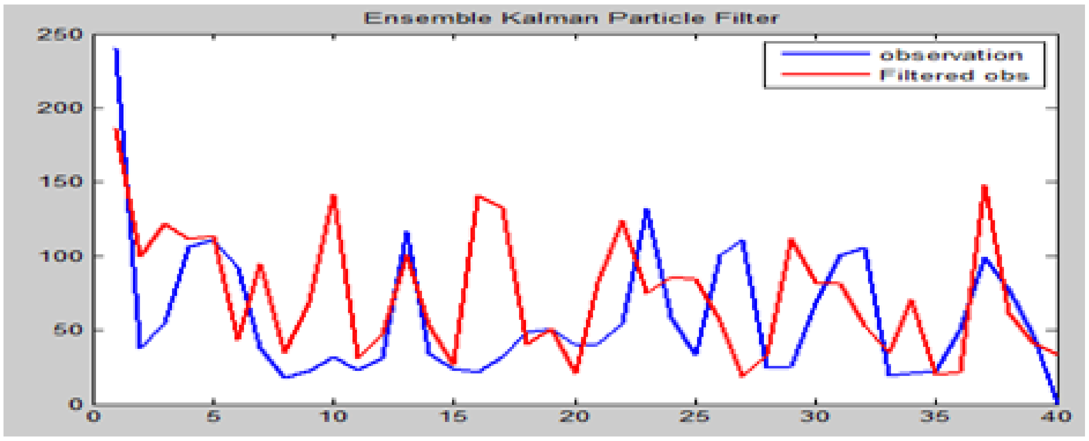

Figure 23.3: Observations for EnKPF under random walk

8.7.3. Root mean square error for the Filters under quasi-Gaussian with random walk

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	3.3245	3.0175	0.84663

Table 11 : Filters under quasi-Gaussian with random walk

8.8 Simulations of filters under quasi-Gaussian with Laplacian Noise

The quasi-Gaussian with Laplacian noise is discussed in the chapter 7 in section 7.4. The Matlab source code is provided in the appendix.

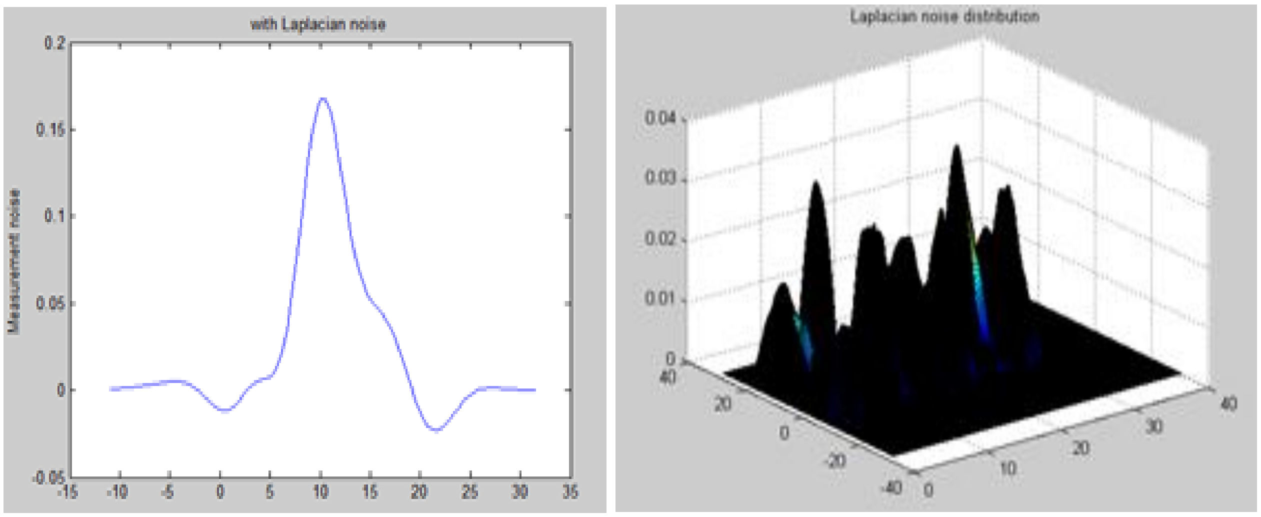

Figure 25: Quasi-Gaussian with Laplacian Noise

8.8.1 State and estimated state of the filters under Laplacian noise

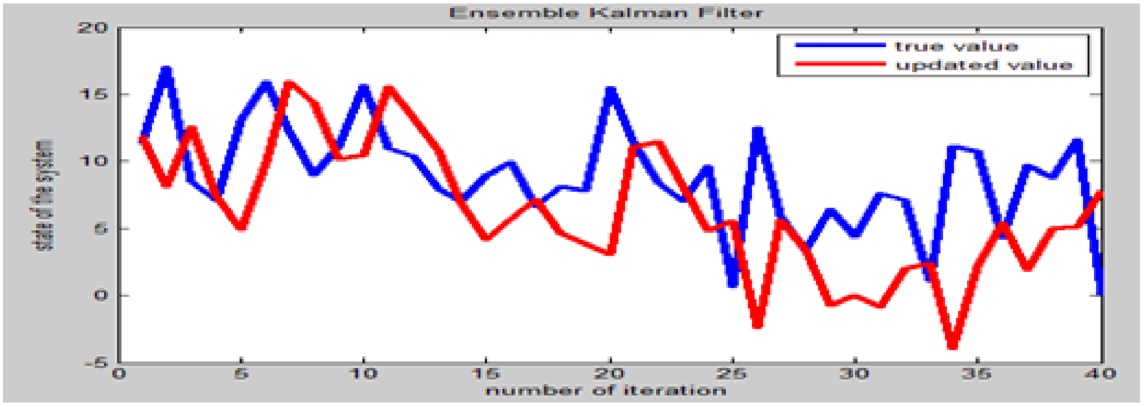

Figure 26.1: State and estimated state for EnKF under Laplacian noise

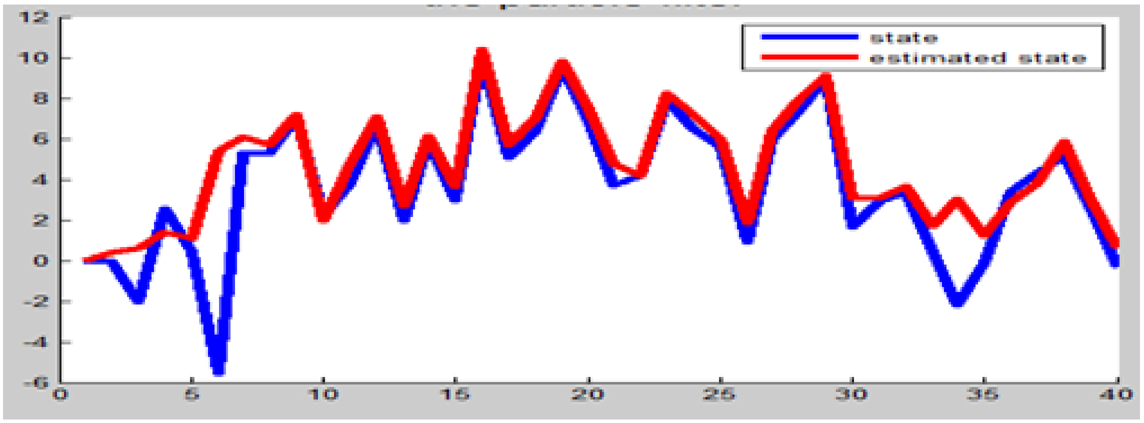

Figure 26.2 : State and estimated state for PF under Laplacian noise

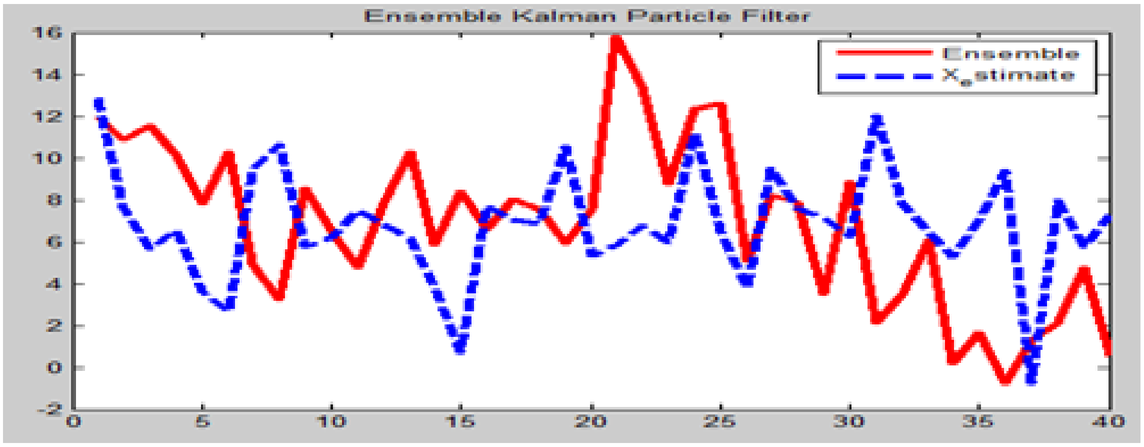

Figure 26.3 : State and estimated state for EnKPF under Laplacian noise

8.8.2.Observation and filtered observation under Laplacian Noise

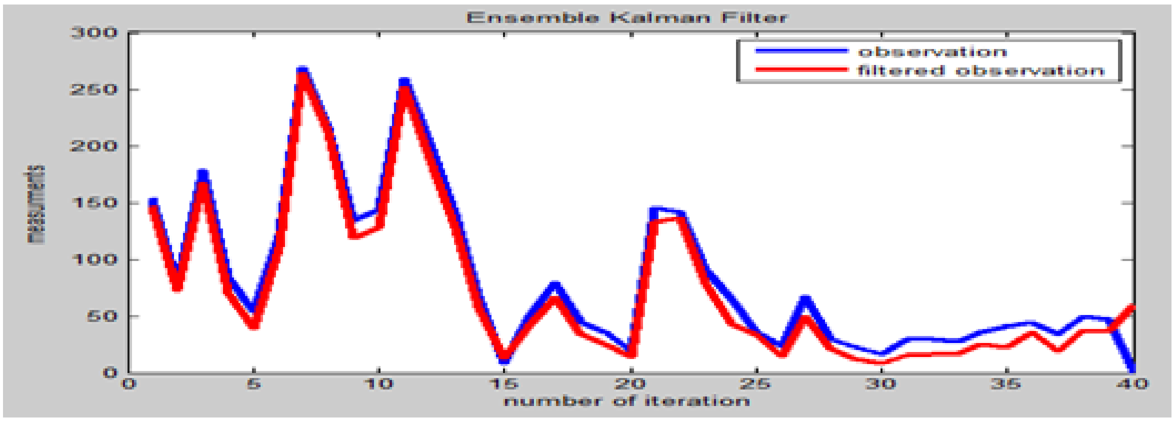

Figure 27.1: Observation for EnKF under Laplacian Noise

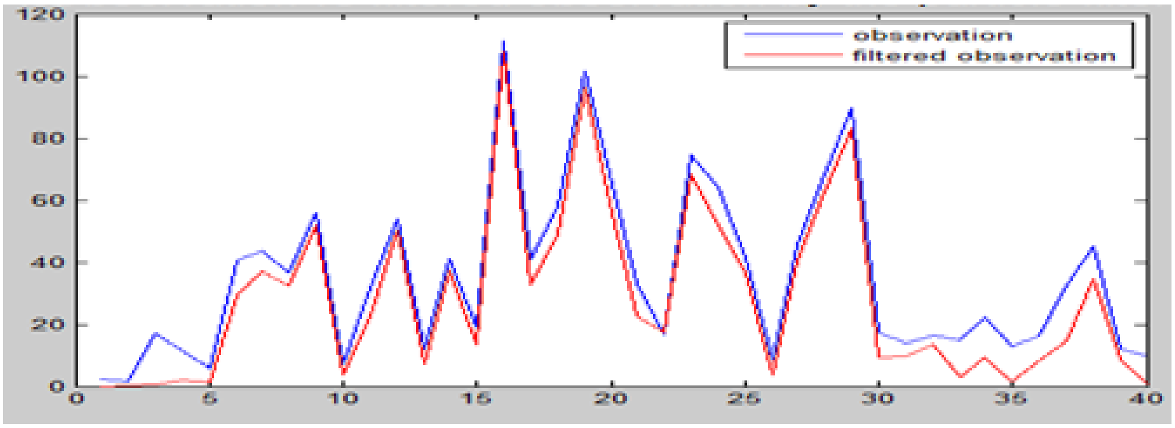

Figure 27.2 : Observation for PF under Laplacian Noise

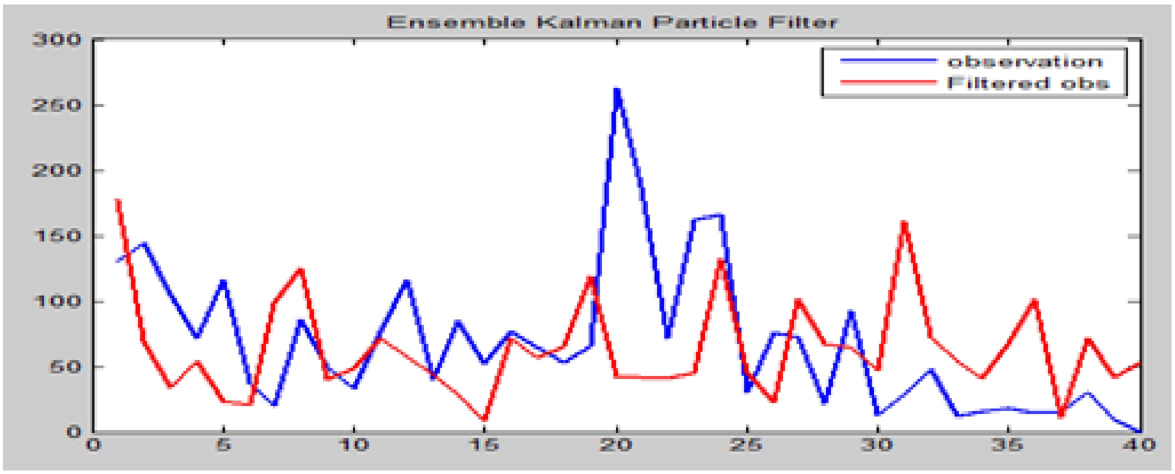

Figure 27.3: Observation for EnKPF under Laplacian Noise

8.8.3. Root mean square error for the Filters under quasi-Gaussian with Laplacian noise

Filters	**EnKF**	**PF**	**EnKPF**
Norm.rmse	3.0448	1.3075	0.9422

Table 12 : Filters under quasi-Gaussian with Laplacian noise

By analyzing the result from tables 8,9,10,11,12 its shown that under all the quasi-Gaussian conditions the Ensemble Kalman Particle filter provides minimum root mean square error than the other filters.

Chapter 9

Conclusion and Future work

A lot of progress has been made in the field of estimation in recent years. Still, it is very difficult to address the nonlinear and non-Gaussian observations. These days the Kalman-based filters use Gaussian distribution to approximate the non-Gaussian distribution. This approach is not efficient because it can result in non-trivial estimation errors. In this thesis, the two filters the Ensemble Kalman filter and the Particle filter are described. For the linear model under Gaussian noise conditions the Ensemble Kalman Filter works well and also the root mean square error is less compared with other filters. The main disadvantage of the EnKF is the inability to handle the non-Gaussian posterior distributions. The Particle filter is used but the estimates depend on the finite samples with weights updated by the likelihoods so it gives the impression that the Particle filter can handle all possible noise statistics of the model. The main disadvantage is high cost for the computations used to prevent the filter degeneracy. And also the sample size grows exponentially with the dimension of the system. Hence, the Particle filter is not useful for high dimensional models. In order to handle these kinds of problems the Ensemble Kalman Particle Filter was developed using a bridging strategy to combine both filters (EnKF and PF).

Initially, EnKPF is used for the linear measurement functions and now this filter is extended to nonlinear measurement functions. Therefore this thesis analyzed the filter under different noise conditions and compared it with other filters. While considering the norm of the root mean square error, it is comparatively less for Ensemble Kalman Particle filter for quasi-Gaussian measurements. From these simulation results, the optimum value for gamma which is the tuning parameter to bridge the Ensemble Kalman Filter and Particle filter is 0.2 for the quasi-Gaussian distribution and 0.6 for the Gaussian distribution. When γ close to zero it will work as a Particle Filter and if it is close to 1 it works similar to the Ensemble Kalman Filter.

This work also considered the generation of a quasi-Gaussian noise in order to analyze the Ensemble Kalman Particle filter. To test this filter under non-Gaussian measurement noise, further studies are required such as the study of various non-Gaussian measurement noises in the sensors, resampling methods, formulation of weights, etc. The main concern is the efficiency and performance of this filter when applied to a realistic model.

APPENDIX

The MATLAB^{TM} source code used for the simulation has been presented in this thesis.
For the comparison purpose the system and observation model are same.

The Ensemble Kalman Particle Filter

```
clear all;
close all;
Nens=100; % number of ensemble
T=40;   % number of steps
% system and observation
nx = 1; %number of states
sys = @(j, Xjm1, Uj) Xjm1/2 + sin(1.2*j) + Uj; %Process equation x[k] = sys(k, x[k-1], w[k]);
where xkm1 is the previous state
ny=1; % number of observed state
obs = @(j, Xj, Vj) Xj^2 + Vj; % Observation equation y[k] = obs(k, x[k],    v[k]) similar to h
 % generation of noise
 nu=1;   % size of the vector of process noise
 m=3;% non zero mean
 sigma_u = sqrt(10);
 gen_sys_noise= @(u) normrnd(m, sigma_u);
 nv=1;% size of the vector of observation noise
 mean=0.3;
 R=0.6;
 r_chol=chol(R);
 sigma_v = sqrt(1);
 gen_obs_noise = @(v) normrnd(mean, sigma_v);
% seperate memry space
 X = zeros(nx,T,Nens);  Y = zeros(ny,T,Nens); h_vi=zeros(ny,T,Nens);
 U = zeros(nu,T,Nens);  V = zeros(nv,T,Nens); Xm=zeros(T,nx);Ym=zeros(T,ny);
 E_X=zeros(nx,Nens); wip1=zeros(T,Nens);
 E_Y=zeros(ny,Nens);
 E_h_om=zeros(1,Nens);E_om=zeros(1,Nens);
 Vi=zeros(1,T,Nens);
 e1=zeros(1,T,Nens);
 Om=zeros(T,Nens);
 h_om=zeros(1,T,Nens);
%P_xy=zeors(nx,ny);
g=0.5;
y=zeros(ny,T,Nens);
whil=1;
 % Simulate system
 while(whil)
for j=1:Nens
    X(1,1,j) = 12;                          % initial state
    U(1,1,j) = 0;                         % initial process noise
    V(1,1,j) = gen_obs_noise(sigma_v);          % initial observation noise
    Y(1,1,j) = obs(1, X(1,1,j), V(:,1));
end
%Prediction/Forecast of ensemble members for the state and the
%observations for the next instant
for j=1:T-1

    for k = 1:Nens
        U(1,j,k) = gen_sys_noise();
        V(1,j,k) =gen_obs_noise()+10*k ;
        X(1,j+1,k) = sys(j, X(1,j,k), U(1,j,k));
        Y(1,j,k) = obs(j, X(1,j,k),   V(1,j,k));
        y(1,j,k)=obs(j,X(1,j+1,k),V(1,j,k));
    end
end
for j=1:T-1
    for k=1:Nens
        Xm= (1/Nens)*sum(X(1,j+1,:));
        E_X(:,k)=X(:,j,k)-Xm;
        Ym= (1/Nens)*sum(Y(1,j+1,:));
        E_Y(:,k)=Y(:,j,k)-Ym;
    end
end
    P_xy=(1/(Nens-1))*E_X*E_Y';
    P_yy=(1/(Nens-1))*(E_Y*(E_Y')+(R/g));
    K1_g=P_xy*(inv(P_yy));
 for j=1:T
   for k=1:Nens
         Vi(:,j,k)=X(:,j,k)+K1_g*(y(:,j,k)-Y(:,j,k));
   end
 end
 for j=1:T
     e1=randn(1,Nens)*r_chol;
     Om(j,:)=K1_g*e1/(sqrt(g));
     for i=1:Nens
         OM=Om*Om';
         Q=(1/(Nens-1))*sum(OM(1,j,:));
         Om_m=(1/Nens)*sum(Om(j,:));
         h_om(1,j,i)=obs(j,Om(j,i),V(1,j,i));
         hm_om=(1/Nens)*sum(h_om(:,j,:));
         E_h_om(:,i)=h_om(:,j,i)-hm_om;
         E_om(:,i)=Om(j,i)-Om_m;
     end
 end
 HQHt=(1/(Nens-1))*E_h_om*(E_h_om');
 for j=1:T
     for k=1:Nens
         h_vi(1,j,k)=obs(j,Vi(1,j,k),V(1,j,k));
     end
 end
 for j=1:T
     for k= 1:Nens
        wip1(j,k)=y(:,j,k)-h_vi(:,j,k);
     end
 end
 wim=(1/Nens)*ones(Nens,Nens);
 wi=expm(-(1/2)*(wip1')*(inv(HQHt+(R/(1-g))))*wip1);
 %wi=(1/(sqrt((2*pi)^4)*abs(R/(1-g))))*wi;
 for i=1:Nens
     for j=1:Nens
        wim(i,j)=wi(i,j)/sum(wi(i,:));
     end

 end
 wim=(1/Nens)*wim;
 wk= wi*(inv(wim));
 wk_=(1/Nens)*wk;
 diff=zeros(Nens,1);
 for i=1:Nens
     diff(i,1)=(wi(i,1)-(1/Nens))^2;
 end
    sumd=sum(diff(:,1));
    Neff=(Nens/(1+(Nens*sumd)))*30;
  X_iu=zeros(1,T,Nens);
 Tau=Neff/Nens;
 tau1=0.8;
 tau2=0.9;
 if(Tau> tau1)
     if(Tau<tau2)
         for i=1:T
            indx = randsample(1:Nens,Nens);
            for j=1:Nens
                X_iu(:,i,j)=Vi(:,i,indx(j))+Om(i,j);
            end
         end
        whil=0;
     else
         g=g-0.05;
         whil=1;
     end
 else
     g=g+0.05;
     whil=1;
 end
```

```
 h_Xiu=zeros(1,T,Nens);
for j=1:T

     for k=1:Nens
         h_Xiu(1,j,k)=obs(j,X_iu(1,j,k),V(1,j,k));
     end
 end
     K2gp1=(1/(Nens-1))*(E_om*(E_h_om'));
     K2gp2=(1/(Nens-1))*((E_h_om*(E_h_om)')+(R/(1-g)));
     K2g=K2gp1*(inv(K2gp2));
  X_est=zeros(1,T,Nens);
for i=1:T
    e2=randn(1,Nens)*r_chol;
    for j=1:Nens
        X_est(:,i,j)=X_iu(:,i,j)+K2g*(y(:,i,j)+(e2(:,j)*(1/sqrt(1-g)))-h_Xiu(:,i,j));
    end
end
 end % end for the while loop
 y_Xest=zeros(1,T,Nens);
for j=1:T

     for k=1:Nens
         y_Xest(1,j,k)=obs(j,X_est(1,j,k),V(1,j,k));
     end
end
X_diff=zeros(1,T,Nens);
% Xtrue=squeeze(X(:,:,Nens));
% X_es=squeeze(X_est(:,:,Nens));

 for j=1:T
     for k=1:Nens
         X_diff(:,j,k)= X_est(:,j,k)-X(:,j,k);
     end
 end
 X_dif=squeeze(X_diff(1,:,:));
rms_X=chol((X_dif)*X_dif');
```

The Ensemble Kalman Filter

The system and observation model are same as the EnKPF

```
% seperate memry space
 X = zeros(nx,T,Nens);  Y = zeros(ny,T,Nens); h_vi=zeros(ny,T,Nens);
 U = zeros(nu,T,Nens);  V = zeros(nv,T,Nens); Xm=zeros(T,nx);Ym=zeros(T,ny);
 E_X=zeros(nx,Nens); wip1=zeros(T,Nens);
 E_Y=zeros(ny,Nens);
 E_h_om=zeros(1,Nens);E_om=zeros(1,Nens);
 Vi=zeros(1,T,Nens);
 e1=zeros(1,T,Nens);
 Om=zeros(T,Nens);
 h_om=zeros(1,T,Nens);
%P_xy=zeors(nx,ny);
g=0.5;
y=zeros(ny,T,Nens);
Y_fil=zeros(1,T,Nens);
 % Simulate system
for j=1:Nens
   X(1,1,j) = 12;                      % initial state
   U(1,1,j) = 0;                     % initial process noise
   V(1,1,j) = gen_obs_noise(sigma_v);        % initial observation noise
   Y(1,1,j) = obs(1, X(1,1,j), V(:,1));
end
%Prediction/Forecast of ensemble members for the state and the
%observations for the next instant
for j=1:T-1

   for k = 1:Nens
      U(1,j,k) = gen_sys_noise();
      V(1,j,k) =gen_obs_noise() ;
      X(1,j+1,k) = sys(j, X(1,j,k), U(1,j,k));
      Y(1,j,k) = obs(j, X(1,j,k),   V(1,j,k));
      y(1,j,k)=obs(j,X(1,j+1,k),V(1,j,k));
   end
end
f=squeeze(V(1,:,:));
R_v=cov(f');

 r=[2,5];
 mu=rand(1,T)*range(r)+min(r);
 vi=mvnrnd(mu,R_v,Nens);
Yi=(squeeze(Y(1,:,:)))+vi';
h_xm=zeros(T,1);
for j=1:T-1
   for k=1:Nens
```

```
        Xm(j,1)= (1/Nens)*sum(X(1,j+1,:));
        h_xm(j,1)=obs(j,Xm(j,1),V(1,j,k));
        E_X(:,k)=X(:,j,k)-Xm(j,1);
        Ym= (1/Nens)*sum(Y(1,j+1,:));
        E_Y(:,k)=Y(:,j,k)-h_xm(j,1);
        p_xy1=sum((X(:,j,k)-Xm(j,1))*(Y(:,j,k)-h_xm(j,1))');
        p_yy1=sum((Y(:,j,k)-h_xm(j,1))*(Y(:,j,k)-h_xm(j,1))');
    end
end


    P_xy=(1/(Nens-1))*p_xy1;
    P_yy=((1/(Nens-1))*p_yy1)+R_v;
    K=P_xy*(inv(P_yy));
    X_up=[squeeze(X(1,:,:))]+[K*(Yi-(squeeze(Y(1,:,:))))];
    X_tr=zeros(T,Nens);
    Y_tr=zeros(T,Nens);
    for i=1: T
        for j= 1:Nens
            X_tr(i,:)=Xm(i,1)+U(:,i,j);
            Y_tr(i,:)=h_xm(i,1)+V(:,i,j);

        end
    end
    for i= 1:T
        for j= 1:Nens
            Y_fil(1,i,j)= obs(j, X_up(i,j),V(1,i,j));
        end
    end
```

The particle filter

```
clear, clc, close all;
nx = 1;
sys = @(k, xkm1, uk) xkm1/2 + sin(1.2*k) + uk;
ny = 1;
obs = @(k, xk, vk) xk^2 + vk;
nu = 1;
sigma_u = sqrt(10);
p_sys_noise   = @(u) normpdf(u, 0, sigma_u);
gen_sys_noise = @(u) normrnd(0, sigma_u);
nv = 1;
sigma_v = sqrt(1);
p_obs_noise   = @(v) normpdf(v, 0, sigma_v);
gen_obs_noise = @(v) normrnd(0, sigma_v);
gen_x0 = @(x) normrnd(0, sqrt(10));
p_yk_given_xk = @(k, yk, xk) p_obs_noise(yk - obs(k, xk, 0));
T = 40;
x = zeros(nx,T);  y = zeros(ny,T);
u = zeros(nu,T);  v = zeros(nv,T);
xh0 = 0;
u(:,1) = 0;
v(:,1) = gen_obs_noise(sigma_v);
x(:,1) = xh0;
y(:,1) = obs(1, xh0, v(:,1));
for k = 2:T

   u(:,k) = gen_sys_noise();
   v(:,k) = gen_obs_noise();
   x(:,k) = sys(k, x(:,k-1), u(:,k));
   y(:,k) = obs(k, x(:,k),   v(:,k));
end
xh = zeros(nx, T); xh(:,1) = xh0;
yh = zeros(ny, T); yh(:,1) = obs(1, xh0, 0);
pf.k               = 1;
pf.Ns              = 200;
pf.w               = zeros(pf.Ns, T);
pf.particles       = zeros(nx, pf.Ns, T);
pf.gen_x0          = gen_x0;
pf.p_yk_given_xk   = p_yk_given_xk;
pf.gen_sys_noise   = gen_sys_noise;
for k = 2:T
   %fprintf('Iteration = %d/%d\n',k,T);
     pf.k = k;
     [xh(:,k), pf] = particle_filter(sys, y(:,k), pf, 'systematic_resampling');
    % filtered observation
   yh(:,k) = obs(k, xh(:,k), 0);
end
```

```
function [xhk, pf] = particle_filter(sys, yk, pf, resampling_strategy)
k = pf.k;
Ns = pf.Ns;                              % number of particles
nx = size(pf.particles,1);               % number of states

wkm1 = pf.w(:, k-1);                     % weights of last iteration
if k == 2
   for i = 1:Ns                          % simulate initial particles
      pf.particles(:,i,1) = pf.gen_x0(); % at time k=1
   end
   wkm1 = repmat(1/Ns, Ns, 1);           % all particles have the same weight
end
xkm1 = pf.particles(:,:,k-1); % extract particles from last iteration;
xk   = zeros(size(xkm1));     % = zeros(nx,Ns);
wk   = zeros(size(wkm1));     % = zeros(Ns,1);
for i = 1:Ns
    xk(:,i) = sys(k, xkm1(:,i), pf.gen_sys_noise());
    wk(i) = wkm1(i) * pf.p_yk_given_xk(k, yk, xk(:,i));
end;
wk = wk./sum(wk);
Neff = 1/sum(wk.^2);
resample_percentaje = 0.50;
Nt = resample_percentaje*Ns;
if Neff < Nt
[xk, wk] = resample(xk, wk, resampling_strategy);
end
xhk = zeros(nx,1);
for i = 1:Ns;
   xhk = xhk + wk(i)*xk(:,i);
end
pf.w(:,k) = wk;
pf.particles(:,:,k) = xk;

return;
function [xk, wk, idx] = resample(xk, wk, resampling_strategy)
Ns = length(wk);
edges = min([0 cumsum(wk)'],1); % protect against accumulated round-off
      edges(end) = 1;                % get the upper edge exact
      u1 = rand/Ns;
      [~, idx] = histc(u1:1/Ns:1, edges);
xk = xk(:,idx);
wk = repmat(1/Ns, 1, Ns);
return;
```

Laplacian noise

```
Nens=100;
T=40;
x=zeros(1,Nens);
m1=5;
varx=10;
   u=rand(1,Nens);
   for j=1:T
    for i=1:Nens
        if u(1,i)>0.5
            x(1,j,i)=m1+sqrt(varx).*(1/sqrt(2))*log(1/(2*(1-u(1,j,i))));
        else
            x(1,j,i)=msqrt(varx)*(1/sqrt(2))*log(2*u(1,i));
        end
     end
   end
```

Random walk

```
delta_x=0.1; %length of space step
tau=0.01; %length of time step
T=40; %number of time steps
M=tau*T; %end time
Nens=100; %number of particls
pos=zeros(T,Nens); %all particles start at x=0
p_l=.5; %probability of moving left
p_r=.5; %probability of moving right
t=0; %counts passage of time
m1 = 10; % mean  of x
m2 = 4; % variance of x
m3 = 5; % target skewness. NB: m3 is equal to 0 for a gaussian variable
m4 = 6 ; % target kurtosis. NB: m4 is equal to 3 for a gaussian variable
%Simulate the random walk process
while t <= M %do something until a specific condition is met
    p= randn(1,T,Nens);%1xN array of random numbers between 0 and 1
 for j=1:T
    for i=1:Nens %loop through each particle to see if it moves
        if p(j,i) < p_l
            pos(j,i)=pos(j,i)-delta_x; %particle moves left
        elseif p(j,i) < (1-p_r)
            pos(j,i)=pos(j,i); %particle doesn't move
        else
            pos(j,i)=pos(j,i)+delta_x; %particle moves right
        end
    end
    t=t+tau; %update time
 end end
```

References

[1] T. R. Bayes, “Essay towards solving a problem in the doctrine of chances,” Phil. Trans. Roy. Soc. Lond., vol. 53, pp. 370–418, 1763. Reprinted in Biometrika, vol. 45, 1958

[2] J. M. Bernardo and A. F. M. Smith, **Bayesian Theory**, 2nd ed., New York: Wiley, 1998.

[3]C. ZHE, "Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond," Chenbayesian.

[4] S. MacBride, **Biological Learning and Control**. Cambridge, US: MIT Press, 2012.

[5] G. Welch and G. Bishop, "An Introduction to The Kalman Filter," technical report, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, 2002.

[6] M. S. Grewal and A. P. Andrews, **Kalman Filtering Theory and Practice Using MATLAB**, Third ed. New Jersy: A JOHN WILEY & SONS, INC., Publication, 2008.

[7] Arthur Gelb. **Applied Optimal Estimation**. M.I.T. Press, 1974.

[8] T. Lefebvre, H. Bruyninckx and J. De Schutter, "Kalman filters for non-linear systems: A comparison of performance", Int. J. Control, vol. 77, no. 7, pp. 639-653, 2004

[9] R. Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation", Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP), vol. 6, pp. 3461-3464, 2001 [10] E. Wan and R. van der Merwe. The Unscented Kalman Filter. Wiley Publishing, 2001

[11] S. Julier and J. Uhlmann, "A New Extension of the Kalman Filter to Nonlinear Systems," Proc. SPIE, vol. 3068, pp. 182-193, 1997.

[12] The Unscented Particle Filter, Tech Report CUED/F-INFENG/TR-380, Cambridge University Engineering Department, Cambridge, England,Aug,2000

[13] E.A Wan and R. Van der Merwe, “The unscented Kalman filter for nonlinear estimation”, Proc.Symp. Adaptive Syst. Signal Process, Commun .Contr., 2000

[14]G. Evensen, "The ensemble Kalman filter: Theoretical formulation and practical implementation", Ocean Dyn., vol. 53, pp. 343-367, 2003

[15] O. Cappe, S. J. Godsill and E. Moulines, "An overview of existing methods and recent advances in sequential Monte Carlo", Proc. IEEE, vol. 95, no. 5, pp. 899-924, 2007

[16] M. Sanjeev Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, 2002

[17] Z. Shen and Y. Tang, "A modified ensemble Kalman particle filter for non-Gaussian systems with nonlinear measurement functions", J. Adv. Model. Earth Syst., vol. 7, no. 1, pp. 50-66, 2015

[18]M. Frei and H. R. Kunsch, "Bridging the ensemble Kalman and particle filter", ArXiv e-prints, 2012

[19]R. E. Kalman, "A new approach to linear filtering and prediction problems", Trans. ASME J. Basic Eng., vol. 82, pp. 34-45, 1960

[20]P. Del Moral, "Measure valued processes and interacting particle systems. Application to nonlinear filtering problems", Ann. Appl. Probab., vol. 8, no. 2, pp. 438-495, 1998

[21]Butala M D, Yun J, Chen Y, Frazin R A, and Kamalabadi F, 2008, Asymptotic Convergence of the Ensemble Kalman Filter, 978-1-4244-1764-3, ICP IEEE

[22]J. Mandel, L. Cobb and J. D. Beezley, "On the convergence of the ensemble Kalman filter", arXiv: 0901.2951, 2009

[23]Do and Chuong B, "The multivariate Gaussian distribution," Section Notes, Lecture on Machine Learning, CS, vol. 229, 2008.

[24]N. Janosh and R. Felix, "MEMS Inertial Sensors Technology,”

[25]K.Siegrist, "The simple random walk," http://www.math.uah.edu/stat/bernoulli/Walk.html.

[26] "Laplace distribution," in Wikipedia, https://en.wikipedia.org/wiki/Laplace_distribution.

[27]" Kurtosis," in Wikipedia, https://en.wikipedia.org/wiki/Kurtosis.